НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

Дипломна робота

на тему: Створення аграрної системи для аналізу щільності грунту з використанням мови програмування Python

Студент групи ТВ-51 Дичко Євген Сергійович _____

(прізвище, ім'я, по батькові) (підпис)

Керівник роботи доц. Галкін Олександр Володимирович _____

(вчені ступінь та звання, прізвище, ініціали) (підпис)

Кількість сторінок 73

Кількість ілюстрацій 30

Київ - 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему створення аграрної системи для аналізу щільності грунту з використанням мови програмування Python.

Виконав: студент ___4__курсу, групи_ТВ-51_

____Дичко Євген Сергійович_____ _____

(прізвище, ім’я, по батькові) (підпис)

Керівник _доц. Галкін Олександр Володимирович_____ _____

(посада, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____

(посада, вчене звання,, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ - 2019

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.В. Коваль

(підпис)

”___”_____2019р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Дичко Євгену Сергійовичу_____

(прізвище, ім’я, по батькові)

1. Тема роботи створення аграрної системи для аналізу щільності грунту з використанням мови програмування Python.

керівник роботи___ доц. Галкін Олександр Володимирович_____

(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____10 червня 2019 р._____

3. Вихідні дані до роботи __Мова програмування Python, середовище розробки Sublime Text 3

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Проектування системи та розробка архітектури, розробка розширеного функціоналу для авторизованого користувача, наочна візуалізація вхідних даних, створення звіту.

5. Перелік ілюстративного матеріалу

Діаграма прецедентів системи, діаграма переміщення даних, архітектура системи, робота користувача у системі.

6. Дата видачі завдання "01" грудня 2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.18-12.01.19	
2	Розробка архітектури та загальної структури системи	12.01.19-10.02.19	
3.	Розробка структур окремих підсистем	10.02.19-15.02.19	
4.	Програмна реалізація системи	15.02.19-12.03.19	
5.	Оформлення пояснювальної записки	13.03.19-30.05.19	
6.	Захист програмного продукту	17.05.19	
7.	Передзахист	31.05.19	
8.	Захист		

Студент _____ _____Дичко Є.С. _____

(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Галкін О. В._____

(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи було створення програмного продукту для аналізу щільності грунту із використанням мови програмування Python з можливістю розширеного функціоналу для авторизованого користувача, зокрема створення звіту і перегляд всієї інформації в особистому кабінеті.

Для повноцінної роботи системи необхідно мати вхідні дані у вигляді JSON – файлу.

Програма написана на мові програмування Python та з використанням технологій HTML/CSS.

Основна мета програми – надання користувача візуального відображення щільності для наочного бачення ситуації і аналізу даних.

Записка містить 55 сторінок і 30 рисунків

ABSTRACT

The purpose of the work was to create a software for soil density analysis using the Python programming language with the advanced functionality of an authorized user, including creating a report and viewing all information in the personal office.
For complete system operation it is necessary to have input data in the form of a JSON-file.
The program is written in the Python programming language and using

HTML / CSS technologies.
The main purpose of the program is to provide the user with visual density mapping for visual vision and data analysis.
The note contains 55 pages and 30 drawings.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Щільність грунту – маса абсолютно сухого ґрунту в одиниці об'єму непорушеної будови. Виражається у г/см^3

3D – 3 dimensional, тривимірний простір.

2D – 2 dimensional, тривимірний простір.

ВСТУП

Зростаючий інтерес населення до сільського господарства призводить до того, що все більше і більше підприємств зміцнюють свою науково-технічну базу. Особливо хочеться звернути увагу на земляні роботи, зокрема, зрошування землі.

Система буде надавати користувачу повноцінну інформацію про щільність грунту на вибраній ділянці поля та на вказаній глибині, що і було метою розробки. Попередньо, всі дані будуть зібрані на підприємстві і внесені до програми. Користувач зможе побачити схематичне відображення поля в якості таблиці,а також об’ємне зображення ділянки поля на різній глибині.

Система здатна знаходити аномальні ділянки грунту (де щільнільність буде від 0 до 10 і від 90 до 10), а також ділянки грунту, де щільність буде рівна тій, що введе користувач програми. Програма може створювати звіт для користувача у вигляді PDF-файлу. Також система здатна працювати із значними обсягами даних з високою швидкістю.

1 ПОСТАНОВКА ЗАДАЧІ

Сьогодні існує багато систем для відображення різних характеристик грунту. Однією з особливість грунту – є його щільність. Тому в заданій задачі потрібно визначити щільність в кожній точці поля і вивести графік в 2D і 3D моделі.

Основними причинами низького впливу земельного потенціалу в Україні є невласні відносини з землею, недосконалість стратегії, залучення землі до вирощування, недосконалі технології, технологія землі землекористування та сільськогосподарського виробництва, а не навмисна ціна, відмова від науково обґрунтованих систем сільського господарства і, зокрема, поширена нездатність до максимальної ротації, так що відсутність органічних добрив, проектування низьких технологічних рівнів, будівництво та експлуатація

меліоративних систем, недосконалість використання системи та зневоднення добрив та навколишнє середовище, комплексні та рекультиваційні, ерозійні та інші заходи .

Щороку градієнт планети виробляє близько 400 млн. Тонн мінеральних добрив, сотні тисяч тонн пестицидів, які отруюють грунт і накопичуються в рослинах, а потім проходять через трофіку, тобто харчові ланцюги надходять в організм.

Тому що стає гірше якість державної країни. Майже по всій країні спостерігається стійке зниження вмісту гумусу в ґрунті, середній вміст якого становить лише 3,5-3,2%.

Застосування великої кількості добрив, пестицидів та інших хімічних речовин з промисловим забрудненням та випромінюванням ще більш ускладнює екологічну ситуацію в Україні, знижує відтворювальну здатність біосфери та екологічну стійкість сільськогосподарських ландшафтів.

Для виконання поставленої задачі підприємство має надати повну інформацію зі своїми вимірами. Інформація передається з датчика при обробці грунту. Після цього, ця інформація записується в вигляді CSV файлу і передається заданій системі.

2D вигляд – це поле, яке відображається, як координатна площина з віссю X і У. Знаючи щільність в кожній точці, замальовуємо квадрати в відповідний колір – 0% – це білий колір, 100% - це чорний колір. Приклад зображено на рисунку 1.1

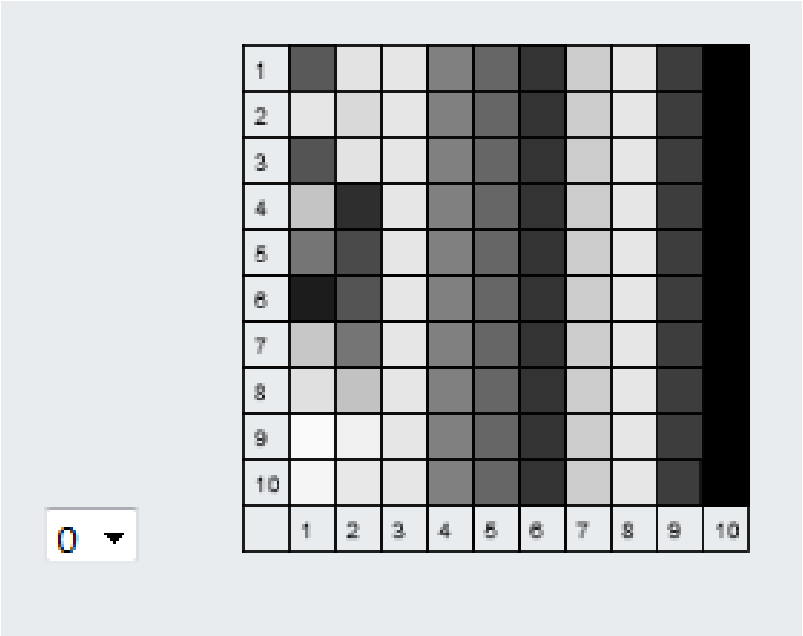


Рисунок 1.1 – 2D зображення візуалізації щільності грунту.

Тривимірна графіка (3D - з англійської тривимірності - "три виміри") працюють з об'єктами в тривимірному просторі. Як правило, результати є плоскою картиною, проекцією. Тривимірна комп'ютерна графіка широко використовується в кіно і комп'ютерних іграх.

У тривимірній комп'ютерній графіці всі об'єкти зазвичай являють собою ряд поверхонь або частинок. Мінімальна площа називається багатокутником. Як багатокутник, зазвичай вибирають трикутники.

Всі візуальні перетворення в 3D-графіці контролюються матрицями. У комп'ютерній графіці використовуються три типи матриць:

- матриця обертання
- переміщення
- масштабування матриці

Кожен багатокутник можна представити у вигляді набору координат його вершин. Тому трикутник має три вершини. Координати кожної вершини - вектор (x, y, z). Якщо перемножити вектор з відповідною матрицею, то

отримуємо новий вектор. Після того, як ми зробили таке перетворення з усіма піками тестового сайту, ми отримаємо новий тестовий сайт, і обертаючи всі полігони, ми отримуємо новий об'єкт, який повертається / зміщується / масштабується відносно оригіналу.

На рисунку 1.2 показано 3D відображення поля для різних глибин.

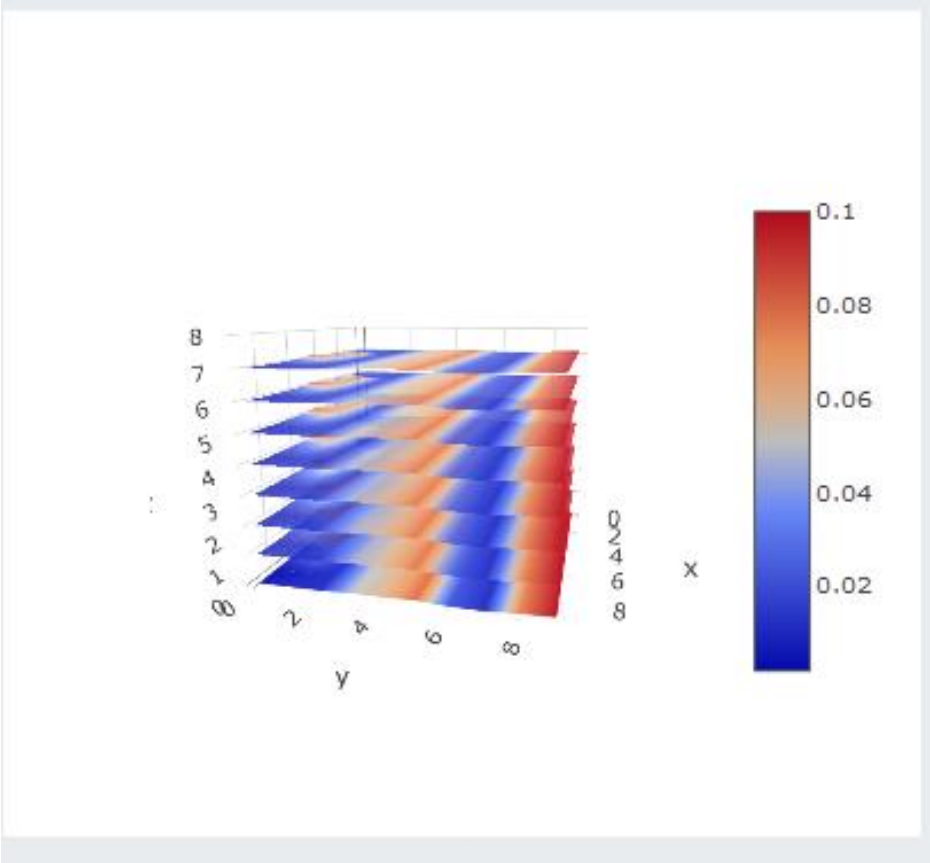


Рисунок 1.2 – 3D зображення візуалізації щільності грунту на різній глибині.

3D вигляд – це об’ємне відображення поля з координатними осями X, У і Z. Програма будує візуальне відображення для глибин від 0 до 8 см.

Користувач матиме змогу ввести вказаний діапазон для щільності грунту і як результат, система знайде всі ділянки поля на різній глибині, яка містить вказану щільність. (рис 1.3)

В РЕЗУЛЬТАТЕ АНАЛИЗА БЫЛИ НАЙДЕНЫ АНОМАЛЬНЫЕ ТОЧКИ:

Глубина	X	Y	Плотность
0	2	0	10
0	7	0	10
0	9	0	100
0	0	1	10
0	2	1	10
0	7	1	10
0	9	1	100
0	2	2	10
0	7	2	10
0	9	2	100
0	2	3	10
0	7	3	10
0	9	3	100
0	2	4	10
0	7	4	10
0	9	4	100
0	2	5	10
0	7	5	10
0	9	5	100
0	2	6	10
0	7	6	10
0	9	6	100
0	2	7	10
0	7	7	10
0	9	7	100

Рисунок 1.3 – Звіт системи для користувача по вхідним даним.

Завдяки можливості створенню звіту, користувач може знаходити аномальні місця на поверхні грунту. Наприклад, ділянки поля з найнижчою щільністю, або навпаки, з найвищою щільністю. На основі даного звіту, користувач може приймати правильні і усвідомленні рішення.

2 МЕТОДИ АНАЛІЗУ ЩІЛЬНОСТІ ГРУНТУ

Загалом сільське господарство в останній час почало дуже різко розвиватись. Все це зв’язано з тим, що родючість українських земель дуже близька до ідеальних. Тому багато іноземних інвесторів мають бажання заволодіти землями на території України

Внаслідок цього створюється дуже багато аграрних систем для спостереженням і аналізом інформації про різні характеристик грунту.

Аналіз даних здійснює послідовні логічні дії для інтерпретації відповідей респондентів та перетворення їх у статистичні форми, необхідні для прийняття маркетингових та управлінських рішень. Зазвичай цей процес складається з трьох послідовних кроків

Щільність грунту відіграє дуже важливу роль в врожайності на даній ділянці землі. Існує дуже багато аналогічних систем, за допомогою яких можна відстежувати щільність грунту на різних ділянках.

Аналіз повинен розглядати не тільки грунт і підземні води, а й кліматичні характеристики території. Зокрема, роль опадів та її кількість. Вони впливають на формування повеней. Інформація, яку ви отримуєте під час аналізу, не є зайвою. Всі вони разом відображають загальну геологічну картину території.

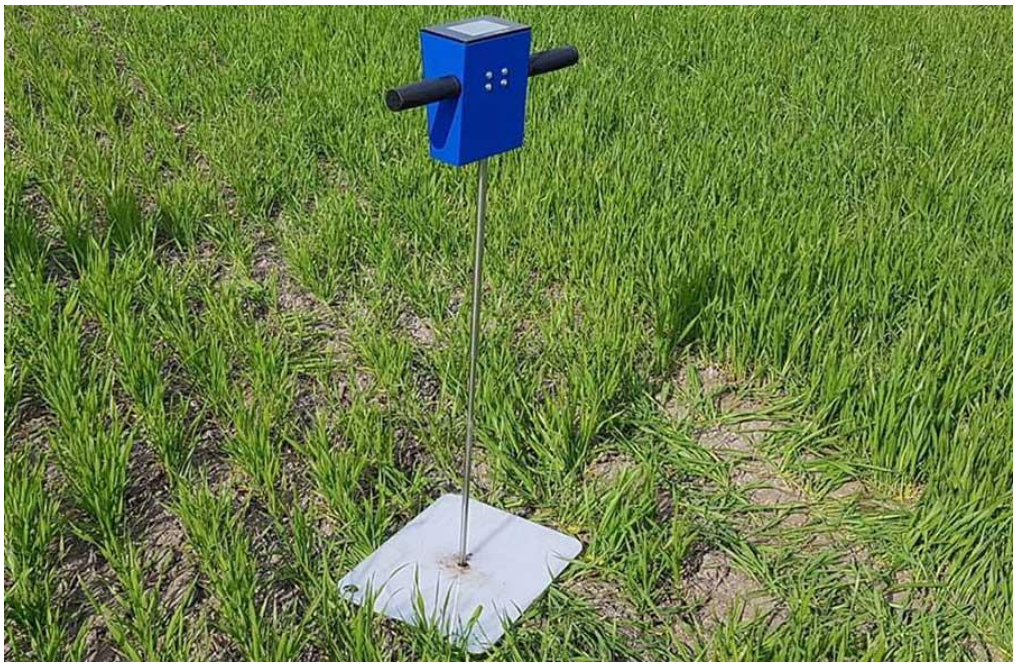


Рисунок 2.1 – Проведення дослідження грунту щільноміром

Можна вивчити різні геологічні елементи. Однак сам грунт є одним з найважливіших елементів. У кожному геологічному дослідженні, перш за все, розглядається склад грунту. Такі дослідження проводяться в місцях, призначених не тільки для будівництва різних напрямків, але і для інших вимог. Такий метод має велике значення для сільськогосподарських земель, наприклад. У таких випадках дуже важливий мінеральний склад грунту і ступінь його забруднення. Нарешті, ця інформація безпосередньо впливає на родючість землі. А це, в свою чергу, найважливіше в сільському господарстві. Іншою метою геологічного обстеження ґрунту є дослідження ступеня забруднення та наявності небезпечних сполук, металів та інших промислових відходів.

Земля є дуже складним об'єктом дослідження. Вона має багато властивостей і хімічних компонентів, які можуть мати позитивну або негативну

фертильність у сільськогосподарському використанні або структурну надійність, коли мова йде про будівництво. Необхідно дуже відповідально проводити аналіз грунту для точного визначення всіх необхідних параметрів. Ці знання допоможуть досягти результатів при роботі на землі, незалежно від мети.

Рисунок 2.2 – Огляд щільноміру.

Метою таких робіт, як аналіз ґрунту, є отримання максимальної інформації про тип ґрунту, його пластичність, пористість і його схильність до пученню під час сильних морозів. Такі дані є дуже цінними. Виходячи з цього, робляться висновки щодо того, який фундамент закласти. Точність даних, отриманих при аналізі ґрунту, залежить від якості та надійності майбутніх будівель або споруд, тому не можна не враховувати ці роботи і їх довіру тільки досвідченим фахівцям.

Слід зазначити, що введення даних є складним процесом і часто вимагає залучення спеціально підготовлених фахівців. У фазі планування досліджень доцільно визначити оптимальну (або краще мінімальну) кількість запитань у анкеті, щоб уникнути труднощів при введенні та аналізі даних. Проблеми погіршуються, коли необхідно обробити велику кількість ділянок полів.

Щільність грунту - важлива характеристика, що показує, в яких умовах ростуть і розвиваються рослини. Від щільності ґрунту залежать всі грунтові

режими: повітрообмін, водопроникність, вологоємність, теплоємність, мікробіологічні і окислювально-відновні процеси. Вона впливає на технологічні властивості, якість обробки грунту. Все це відбивається на величині і якості врожаю. При пухкій будові орного шару створюються умови для підвищеного витрачання вологи на випаровування, а при щільному - несприятливі для розвитку коренів рослин.

Значної шкоди ґрунтам наносить деградація, яка проявляється в ущільненні грунту і погіршенні її структури.

Основні причини ущільнення грунту:

— високий ступінь розораності грунтів;
— застосування інтенсивного обробітку грунту;
— недотримання чергування культур в сівозміні;
— недостатня кількість органічних добрив, які вносять в грунт.

Ходові системи засобів механізації в землеробстві мають неоднакові конструктивні параметри, а тому ущільнюють грунт по-різному: гусеничні трактори менше ущільнюють грунт, ніж колісні. Ходові системи тракторів, в яких гусениці мають менший крок, а опорні катки - менш віддалені один від одного, здатні в меншій мірі ущільнювати грунт. Від конструкції шин залежать питомі навантаження на грунт, деформація його при буксируванні, що впливає на ущільнення грунту. Більше ущільнюється грунт на периферії поля.

В результаті випадання великої кількості опадів ущільнення грунту збільшується через збільшення його маси. Зрошення ущільнених грунтів неефективно, оскільки нерідко призводить до цементації поверхні. Після підсихання на ній утворюються величезні тріщини.

Під час ущільнення грунту відбувається:

— збільшення питомої маси грунту;
— зниження загальної і особливо некапілярної пористості;

- уповільнення зростання кореневої системи - зменшується загальна маса коренів і утруднюється проникнення коренів в орні і підорного шари грунту;
- зменшення вологозабезпечення рослин;
- погіршення водно-фізичних властивостей: влагоемкости, швидкості вбирання поливної води, зменшення водопроникності;
- погіршення аерації і біологічних процесів;
- посилення поверхневого стоку води і змиву мелкозема;
- погіршення поживного режиму ґрунту;
- зниження врожайності і якості сільгосппродукції.

Висновки ґрунтуються на результатах дослідження. Рекомендації передбачають, як слід обробляти висновки. Надання рекомендацій може включати використання знань, які виходять за рамки результатів. Остання частина заявки, що містить додаткову інформацію, необхідна для кращого розуміння результатів. Крім написання звіту для клієнтів, дослідники можуть використовувати усні презентації методів дослідження та результати. У цьому випадку можна відповісти на питання і обговорити отримані дані.

Було проаналізовано багато систем, які схожі за функціоналом до заданої і вибрано певний алгоритм роботи.

Висновки до розділу:

У цьому розділі було повністю описано мету дипломної роботи: аналіз щільності грунту. Було проаналізовано всі можливі варіанти для отримання даних і взаємодію даних з системою.

3 ЗАСОБИ РОЗРОБКИ

Засоби розробки програмного забезпечення, включаючи програмні засоби, необхідні для автоматичного створення машинного коду. Вони є інструментами для професіоналів програмістів і дозволяють розробляти різні програми на різних мовах програмування.

Засоби розробки програмного забезпечення включають наступні програми:

- асемблери – спеціальні комп’ютерні програми, що виконують перетворення вихідного тексту на мові асемблера в машинну команду у вигляді коду;
- транслятори – спеціальні програми, що виконують трансляцію програми;
- компілятори – спеціальні програми, що перетворюють код, написаний на мові високого рівня, в таку ж саму програму на машинній мові;
- інтерпретатори – спеціальні програми, що аналізують оператори програми або команди і відразу ж виконують їх;
- отладчики – спеціальні програми, призначені для пошуку помилок в програмі.

Мови, що представляють алгоритми у вигляді послідовності читаних (Не двійково-кодованих) команд, називаються алгоритмічними мовами. Алгоритмічні мови підрозділяються на машинно-орієнтовані, процедурно-орієнтовані і проблемно-орієнтовані.

Задана система була написана на процедурно-орієнтованій мові.

Процедурно-орієнтовані і проблемно-орієнтовані мови відносяться до мов високого рівня, що використовують макрокоманди.

3.1 Середовище розробки

Основним середовищем розробки було середовище Sublime Text 3.

Sublime Text - це швидкий текстовий редактор між платформами. Підтримуються плагіни, розроблені з використанням мови програмування Python.

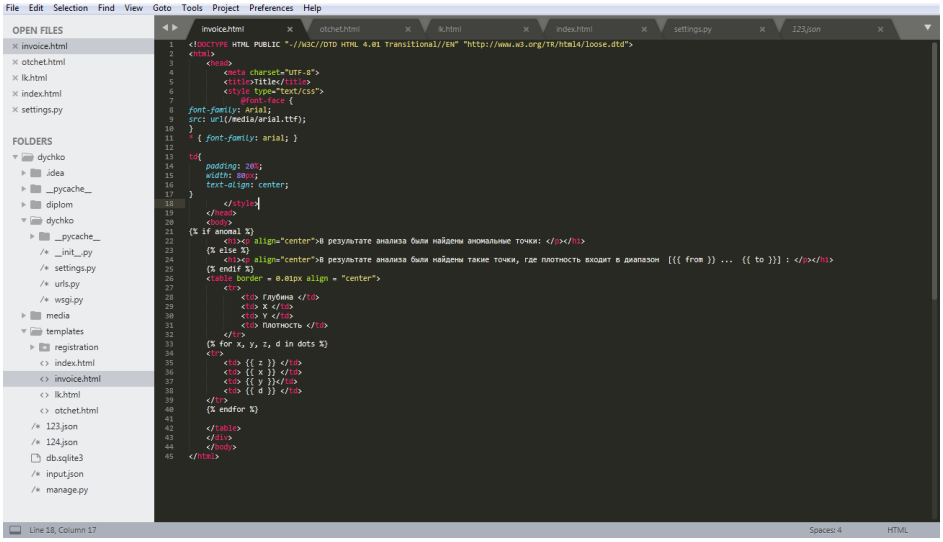


Рисунок 3.1 – інтерфейс Sublime Text 3

Sublime Text не є безкоштовним чи відкритим програмним забезпеченням, але деякі плагіни розповсюджуються за допомогою безкоштовної ліцензії, розробленої та підтримуваної спільнотою розробників.

3.2 Мова програмування та використані технології

Python - об'єктно-орієнтована мова високого рівня зі строгою динамічною типізацією Розроблена в 1990 році Гвідо ван Россумом. Високорівневі структури даних, а також динамічна семантика і динамічні зв'язки роблять його привабливим для швидкої розробки програм, а також для об'єднання існуючих компонентів. Python підтримує модулі і модулі, що полегшує модульність і повторне використання коду. Інтерпретатори та стандартні бібліотеки Python доступні як у складеному, так і у вихідному вигляді на всіх основних платформах. Python мова програмування підтримує кілька парадигм програмування, у тому числі: об'єктно-орієнтовані, процедурні, функціональні та аспектно-орієнтовані.

До основних переваг можна віднести:

• чистий синтаксис (ви повинні використовувати відступи для вибору блоків);

• портативність програм (які є спільними для більшості інтерпретованих мов);

• Стандартний дистрибутив містить велику кількість корисних модулів (включаючи модуль розробки графічного інтерфейсу користувача).

Можливість використання Python в діалоговому режимі (дуже корисно для експериментів і вирішення простих завдань);

• Стандартний розподіл має просте, але потужне середовище розробки, що називається IDLE, написане на Python.

• Практичний для вирішення математичних задач (має засоби для роботи з комплексними числами, може працювати з цілими числами довільного значення, може використовуватися як потужний калькулятор в інтерактивному режимі);

• Відкритий код (можливість редагувати його іншими користувачами).

Python має високорівневі, ефективні структури даних і простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів і той факт, що він є інтерпретованою мовою робить його ідеальним для сценаріїв і швидкого розвитку програм у багатьох галузях на більшості платформ.

Django (Jango) - це відкрита платформа Python високого рівня (програмна основа) для розробки веб-систем. Він був названий на честь джазмена Джанго Рейнхардта (за музичним смаком одного з засновників проекту).

Сайт на Django складається з однієї або декількох частин, модульність яких рекомендується. Це одна з головних архітектурних відмінностей цієї системи від інших (наприклад, Ruby on Rails).

Архітектура Django схожа на контролер модельного перегляду (MVC). Однак те, що називається "контролером" в класичній моделі MVC, називається в Django "типи" (View), але це буде "view", зване "pattern" (Template). Це те, що MVC розробники називають Django MTV (Model Template View).

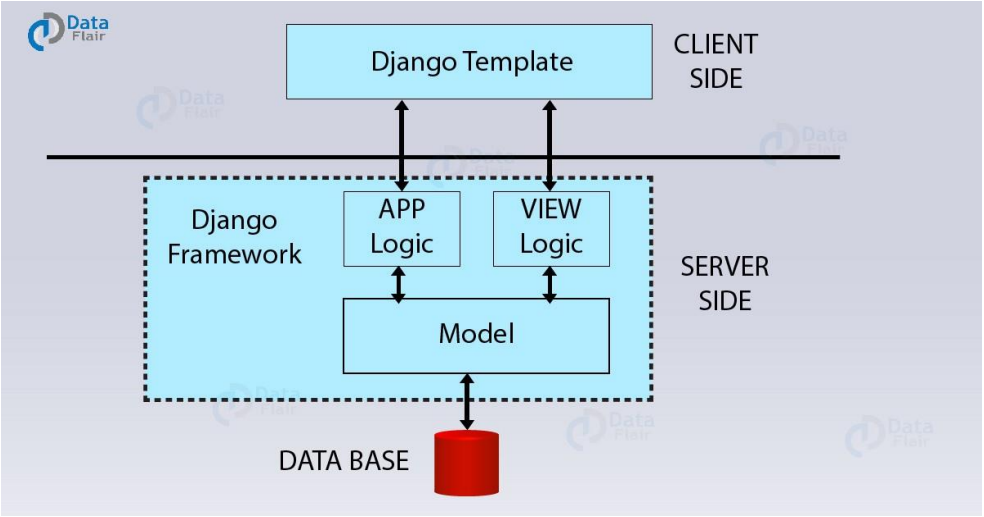


Рисунок 3.2 – Схематичне зображення принципу MTV

Початкова розробка Django, як засобу для роботи новинних ресурсів, досить сильно позначилася на його архітектурі: він надає ряд засобів, які допомагають у швидкій розробці веб-сайтів інформаційного характеру. Наприклад, розробнику не потрібно створювати сторінки для контролерів і адміністративної частини сайту, Django побудований для керування вмістом, який може міститися на будь-якому сайті, створеному на Django, і що кілька сайтів одночасно керують одним сервером . Модуль управління дозволяє створювати, змінювати і видаляти всі об'єкти, зміст сайту, протоколюючи всі дії і надає інтерфейс для користувачів і груп (з правом делегування) керування.

Мова розмітки гіпертексту (HTML) - це стандартна мова розмітки для створення веб-сторінок і веб-додатків. За допомогою Cascading Style Sheets (CSS) і JavaScript вона формує тріаду основних технологій для World Wide Web.

HTML є відносно простим набором кодів, що описує структуру документа. За допомогою HTML можна виділити окремі логічні частини тексту (заголовки, абзаци, списки тощо), помістити фотографію або зображення на веб-сторінку і організувати посилання на сторінці для спілкування з іншими документами.

HTML не визначає конкретні та точні параметри форматування тексту документа. Конкретний тип документа визначає лише браузер на комп'ютері користувача Інтернету.

HTML не є мовою програмування, але веб-сторінки можуть включати вбудовані сценарії в Javascript і Visual Basic Script і аплети в Java.

Веб-браузери отримують HTML-документи з веб-сервера або локальне зберігання та передають документи на мультимедійні веб-сторінки. HTML описує структуру сайту семантично і спочатку містить сигнали для зовнішнього вигляду документа.

Рисунок 3.3 – Огляд можливих браузерів.

Елементи HTML - це блоки HTML-сторінок. З HTML-конструкціями, зображеннями та іншими об'єктами, наприклад, наприклад, інтерактивні форми можуть бути вбудовані у візуалізовану сторінку. HTML надає інструменти для створення структурованих документів, які ідентифікують структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML визначаються тегами, записаними в кутових дужках. Теги, наприклад, безпосередньо вставляння вмісту на сторінку. Інші теги, такі як <img /> <input /> <p>, оточують і надають інформацію про текст документа і можуть містити інші теги, ніж під-елементи. Браузери не відображають HTML-теги, але вони використовуються для інтерпретації вмісту сторінки.

Браузер - програмне забезпечення для комп'ютера або іншого електронного пристрою, який зазвичай підключається до Інтернету і дозволяє користувачеві взаємодіяти з текстом, зображеннями або іншою інформацією на гіпертекстовій веб-сторінці. Тексти та зображення можуть містити посилання на інші веб-сайти, розміщені на цьому ж веб-сайті або на інших веб-сайтах. За допомогою гіперпосилань браузер дозволяє користувачеві швидко та легко отримувати інформацію на багатьох сайтах.

CSS (Cascading Style Sheets) - це спеціальні правила для відображення конкретного елемента в документі HTML, який називається Cascading Style Sheets. Специфікація HTML дозволяє використовувати кілька правил стилю для того самого елемента, який послідовно інтерпретується браузером, тобто каскадами. Формат написання правил CSS схожий на табличне представлення даних.

Заголовок таблиці - це назва елемента, класу або ідентифікатора стилю. Клітини та рядки таблиці - це властивості стилю та їх значення. Під стилем зазвичай розуміється приведення явища до загального набору правил і визначень. Таким чином, CSS - це спосіб додатково форматувати стандартні HTML-теги. HTML версія 4.01 містить такі специфікації CSS:

Основною концепцією CSS є стиль, тобто набір правил дизайну і форматування, які можуть бути застосовані до різних елементів сторінки. У стандартному HTML ці властивості повинні бути описані кожного разу, щоб призначити певні елементи певним елементам (наприклад, колір, розмір, позиція на сторінці тощо), навіть якщо 10 або 110 таких елементів були активовані. відрізняються один від одного. Ви повинні були вставити один і той же шматок HTML-коду десять або сто разів на сторінку, щоб збільшити розмір файлу та час завантаження користувача, який його переглядає.

CSS працює більш комфортно і економічно. Щоб привласнити елемент певним характеристикам, ви повинні один раз описати цей елемент і визначити його як стиль. Тоді все, що вам потрібно зробити, це вказати, що елемент, який ви хочете належним чином сформувати, повинен приймати характеристики описаного стилю.

Крім того, ви можете зберегти опис стилю не в тексті вашої сторінки, а в окремому файлі. Це дозволяє використовувати опис стилю на будь-якій кількості веб-сторінок і змінювати дизайн будь-якої кількості сторінок. Виправте лише опис стилю у (окремому) файлі.

Крім того, CSS дозволяє працювати з візуалізацією сторінок на набагато більш високому рівні, ніж стандартний HTML, уникаючи при цьому необхідності зайвого зважування графіки.

CSS використовується разом з HTML файлом і може бути підключеним до HTML різними шляхами:

— написати стиль в файлі html;

— написати стиль окремо у кожному елементі;

— написати окремий файл зі стилем, який можна підключити до файлу html.

Canvas - це елемент HTML5, який можна використовувати для створення двовимірного растрового зображення за допомогою сценаріїв, як правило, у JavaScript. Початкова точка блоку розташована у верхньому лівому кутку. З цього і будують кожен елемент блоку . Розмір координатного простору не обов'язково відповідає розміру фактично відображуваної області. За замовчуванням ширина складає три сотні пікселів, а висота - сто п'ятдесят.

Зазвичай він використовується для малювання графіки для статей і ігрового поля в деяких браузерних іграх. Він також може бути використаний для вбудовування відео на сторінку і створення повноцінного гравця.

Plotly використовується для створення 3D об’єктів.

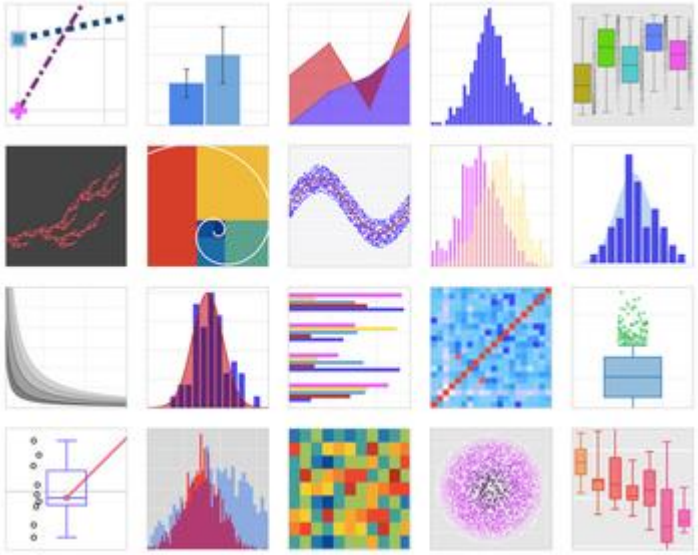


Рисунок 3.4 – Галерея графіків Plotly

Bootstrap - це безкоштовний набір інструментів для створення веб-сайтів і веб-додатків. Містить шаблони оформлення HTML та CSS для друкарства, веб-форм, кнопок, тегів, блоків навігації та інших компонентів веб-інтерфейсу, включаючи розширення JavaScript.

Оскільки Bootstrap використовує сучасні розробки в CSS і HTML, необхідно подбати про підтримку старих браузерів.

Bootstrap 4 стала дуже популярною внаслідок різноманітних факторів, особливо завдяки використанню класів для швидкого створення 12-сторінкової адаптивної сітки. Bootstrap також має ряд плагінів jQuery, які дозволяють вставляти інтерактивні елементи, такі як карусель (або слайдер), підказки, модальні вікна тощо, просто додаючи код.

Bootstrap дуже легкий у використанні і значно полегшує час для написання інтерфейсу користувача. Оскільки дуже багато різних оформлень можна підключити завдяки Bootstrap лише використавши потрібний клас.

Основні засоби Bootsrap:

1) Сітки - це попередньо визначені розміри стовпців, які можна використовувати негайно. Наприклад, ширина стовпця 140 px - це клас.
2) Шаблони - фіксований або гумовий шаблон документа.

3) Типографія - опис шрифтів, визначення певних класів для шрифтів, таких як код, котирування тощо.
4) Медіа - забезпечує керування деякими зображеннями та відео.
5) Столи - це засоби дизайну столу, аж до додаткових функцій сортування.
6) Форми - Формує класи і деякі події, які відбуваються з ними.
7) Навігація - Проектування класів для панелей, вкладок, навігації по сторінках, меню і панелях інструментів.
8) Сповіщення - Створюйте діалогові вікна, підказки та спливаючі вікна.

Висновки до розділу

Розроблена система працює на будь-якій ОС, оскільки її відображення відбувається в браузері. Для повноцінного використання система необхідно мати доступ до інтернету і будь-який браузер. Щоб отримати звіт, користувачу необхідно мати певну програму для роботи з PDF-файлом.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення програмного продукту було використано клієнт-серверну архітектуру, оскільки програма взаємодіє з сервером, де міститься всі дані підприємства і з клієнтом, який може отримати візуальне відображення даних.

Дуже важливо чітко визначити, хто або що вважається "клієнтом". Ви можете розмовляти через клієнтський комп'ютер, який використовується для розмови з іншими комп'ютерами. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, використовуючи відповідне програмне та апаратне забезпечення, ви можете спілкуватися з людьми, які хочуть отримати доступ до тієї чи іншої інформації.

Загальновизнано, що клієнти і сервери є в основному програмними модулями. В основному вони знаходяться на різних комп'ютерах, але є ситуації, коли обидві програми - клієнт і сервер - фізично розташовані на одному комп'ютері. У цій ситуації сервер часто називають локальним.

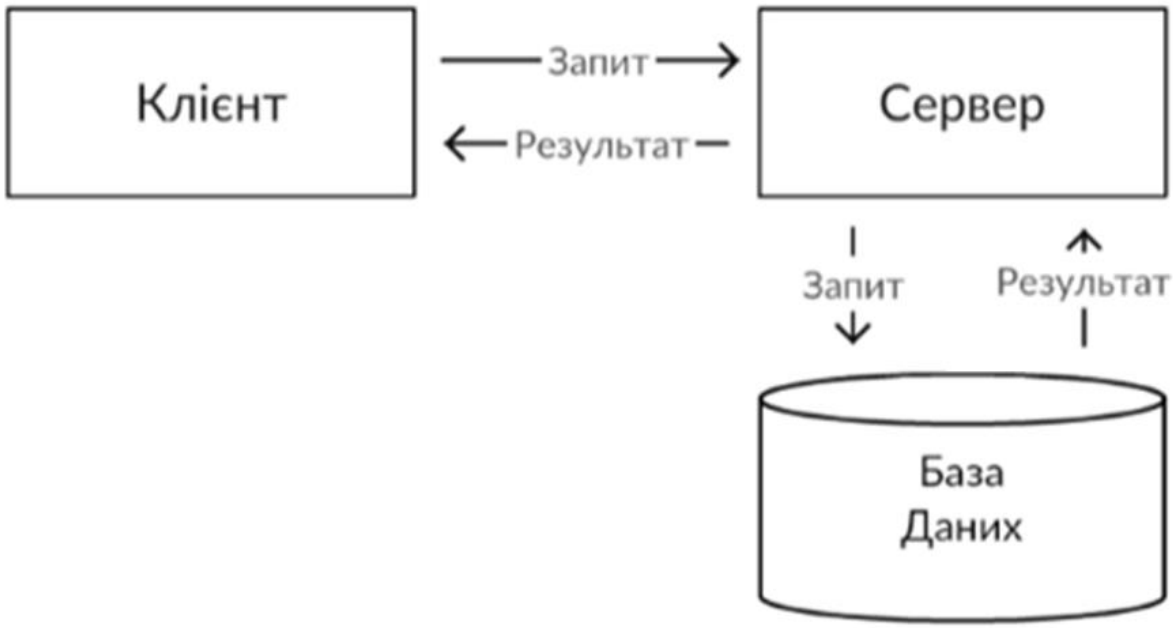


Рисунок 4.1 – Клієнт-серверна архітектура системи

Для написання системи було використано широкий функціонал технологій. Основні актори програми: користувач і адміністратор.

Зручно представити весь функціонал системи завдяки діаграм прецендентів. Осклільки вона повністю охоплює весь функціонал системи і всіх акторів.

Рисунок 4.2 – Діаграма прецендентів

Надалі розглянемо весь функціонал системи.

4.1 Отримання даних

Для створення системи головним було продумати правильний алгоритм для отримання даних. Отримання даних поділяється на три етапи:

- проведення дослідження на ділянці поля з використанням такого приладу, як щільномір;
- перенесення всіх даних після дослідження в таблицю формату .xls або .csv;

— створення json-файлу по даним з таблиці.

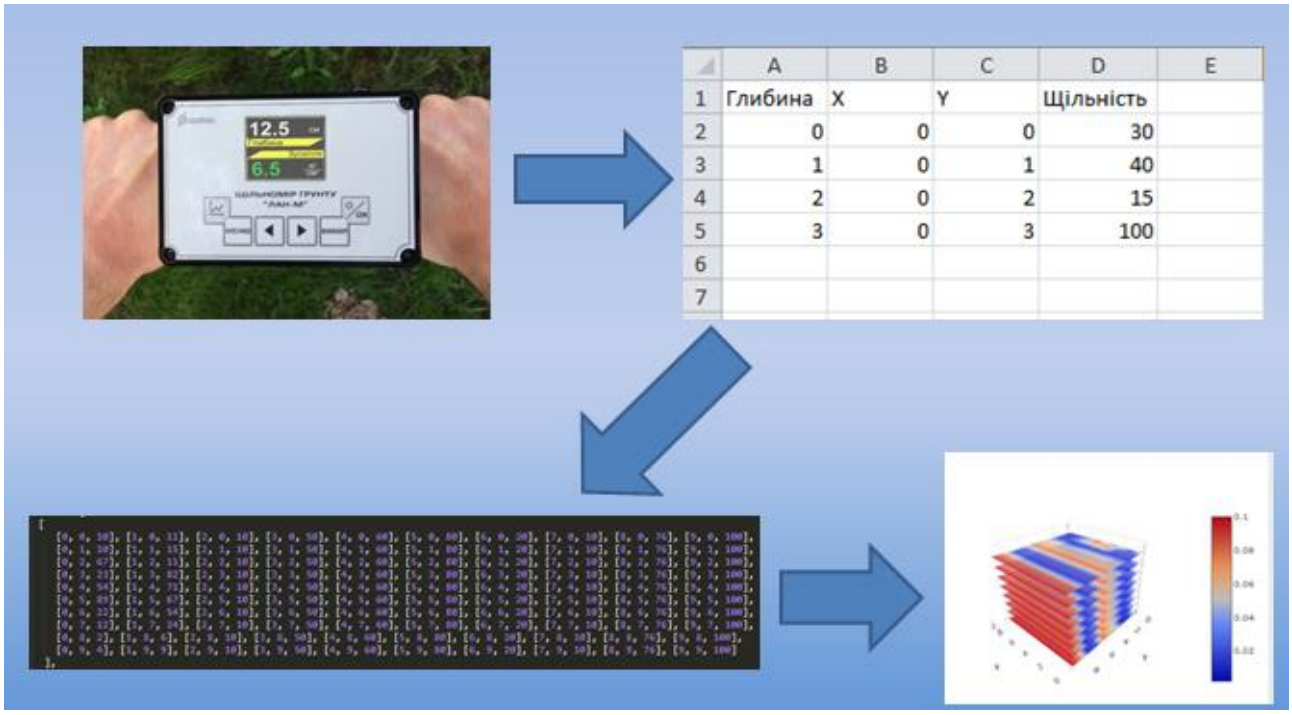


Рисунок 4.3 – Діаграма переміщення даних

JSON (англійська нотація об'єкта JavaScript, написання об'єктів JavaScript, вимовляється Jason) - це текстовий формат, який використовується для обміну даними між комп'ютерами. JSON є текстовою і може бути прочитана однією особою. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат використовується в основному для передачі структурованої інформації по мережі (завдяки процесу, званому серіалізацією).

Розроблено і популяризовано формат завдяки Дугласу Крекфорду.

JSON знайшов свою основну мету у написанні веб-додатків, особливо при використанні технології AJAX. JSON, що використовується в AJAX, діє як заміна XML (використовується в AJAX) в асинхронній передачі структурованої інформації між клієнтом і сервером. Перевага JSON над XML полягає в тому, що

вона дозволяє створювати складні структури в атрибутах, займає менше місця і інтерпретується безпосередньо з JavaScript в об'єктах.

Дуже важливо правильно заповнити JSON – файл, щоб система могла коректно зчитати дані і візуально представити їх. Якщо дані будуть представлені не в правильному порядку – система не зможе правильно зобразити дані для подальшого аналізу.

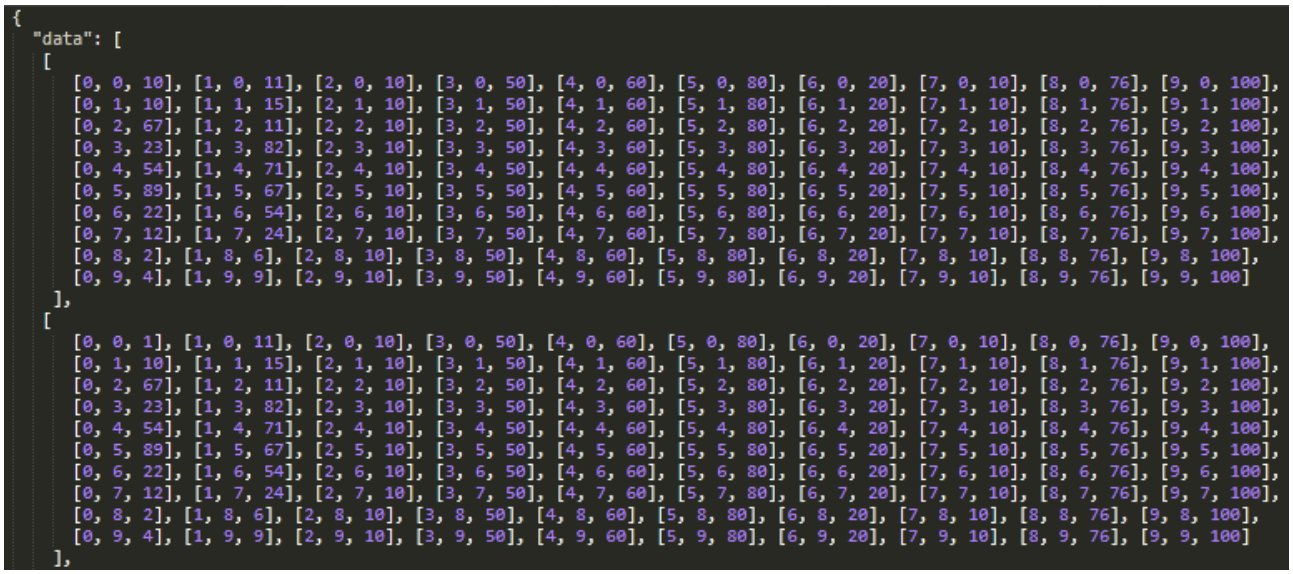


Рисунок 4.4 – Приклад заповнення JSON-файлу

4.2 Реєстрація і авторизація

Оскільки основними акторами системи є адміністратор і користувач. Адміністратор необхідний для реєстрації нового користувача. Зазначимо, що самостійно користувач не має змоги зареєструватися. Оскільки система була створена завдяки технології Django, тому панель адміністратора має дуже зручний вигляд.

Щоденний алгоритм роботи з інтерфейсом керування Django виглядає так: "Виберіть об'єкт і змініть його". Він підходить для більшості випадків. Однак, якщо необхідно зробити ту ж саму дію на безліч об'єктів, така поведінка інтерфейсу буде повільною.

У таких випадках ви можете використовувати інтерфейс керування Django для створення та реєстрації дій - прості функції, які викликаються для виконання конкретних дій у списку об'єктів, виділених на сторінці інтерфейсу.

Завдяки сторінці адміністратора, яку надає технологія Django, можна дуже просто використати CRUD систему, тобто створення, редагування і видалення нових користувачів. Саме тому було вибрано технологію Django для створення даної системи, через її простоту у використанні. Як приклад, Django самостійно надає сторінку адміністратора.

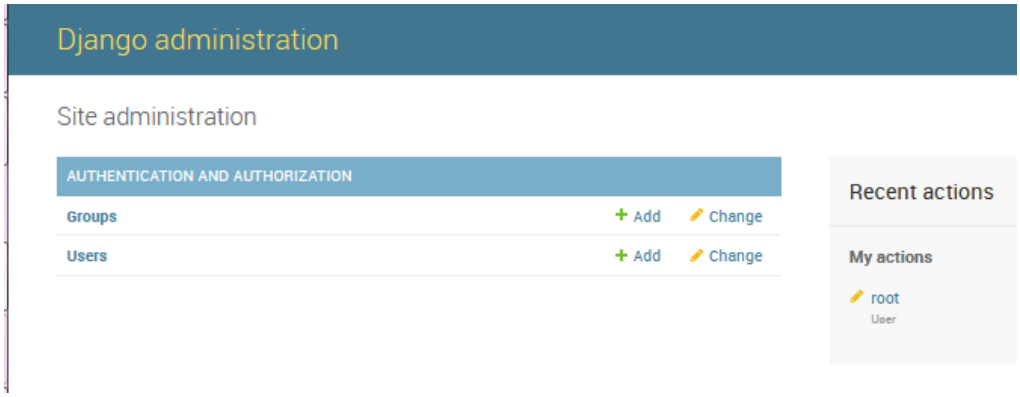


Рисунок 4.5 – Сторінка адміністратора.

Вхід на сайт - це процес повідомлення веб-сайту персональних даних для подальшого доступу до додаткових послуг або доступу до ресурсів, які не можуть переглядати або завантажувати неавторизовані користувачі. Процес реєстрації зазвичай здійснюється шляхом надання ідентифікатора користувача ("логін" (тісно супроводжує логін - ім'я користувача для входу) і пароль - деякі конфіденційні відомості в майбутньому будуть отримувати логін і пароль користувача, система порівнює його з значенням в Спеціальна захищена база даних, що зберігається і, у разі успішної аутентифікації аутентифікує користувача і потім вступає в систему для роботи.

Авторизація - управління рівнями та засобами доступу до безпечного ресурсу, як фізичного (доступ до готельних номерів карткою), так і цифрових технологій (наприклад, автоматизованої системи контролю доступу) та

системних ресурсів залежно від ідентифікатора користувача та пароля або спеціальних повноважень (особи) , програма) для виконання певних дій в системі обробки даних.

Авторизація потрібна для того, щоб надати авторизованому користувачу додатковий функціонал:

— створення звіту по аномальним точкам (від 0 до 10 і від 90 до 100) ;
— створення звіту по точкам, діапазон яких задається користувачем;
— додавання власного поля;
— збереження звіту;
— перегляд даних в особистому кабінеті;
— відкриття звіту у файлі формату PDF.

Для авторизації користувач повинен ввести логін і пароль для входу в систему.

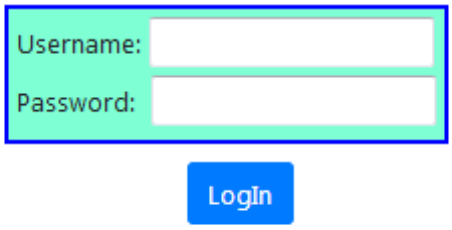


Рисунок 4.6 – Авторизація користувача.

Після проведення авторизації, користувач переходить до головної сторінки.

4.3 Візуалізація даних

Візуалізація даних - це візуальне представлення масивів різної інформації. Існують різні види візуалізації:

— Звичайне представлення в схематичній формі;

— Концептуальна візуалізація;

— Стратегічна візуалізація;

— Комбінована візуалізація.

Візуальна інформація краще сприймається і дозволяє швидко і ефективно передати своїм поглядам і ідеям глядача. Фізіологічно, сприйняття візуальної інформації є фундаментальним для людини. Є численні дослідження, які підтверджують, що:

• 90% інформації, яку люди сприймають через бачення;

• На увазі 70% сенсорних рецепторів;

• Близько половини нейронів людського мозку беруть участь у обробці візуальної інформації;

• На 19% менше робота з візуальними даними використовує когнітивні функції мозку, яка відповідає за обробку та аналіз інформації;

• на 17% вища робота особи, що працює з візуальною інформацією;

• 4,5% краще звертатися до детальних деталей візуальної інформації;

• 60 000 разів швидше візуальна інформація, ніж текст сприймається.

Компетентна візуалізація даних надає вашій системі ряд переваг:

— Швидкість прийняття рішень. Простіше і швидше зробити висновок, якщо поглянути на діаграму, в якій один з стовпців або один з точок взаємодії набагато вище, ніж будь-який інший, ніж прокручування кількох сторінок статистики в Google Sheets або Excel.

— Більше приваблює аудиторію. Як я вже сказав, більшість людей сприймають візуальну інформацію краще і запам'ятовують її.

— Висока ефективність читання. Красива, яскрава графіка з чітким повідомленням приверне увагу ваших читачів.

— Краще розуміння даних. Ідеальні звіти зрозумілі не тільки для технічних фахівців, аналітиків і вчених, але й для менеджера з маркетингу або SEO, що дозволяє кожному співробітнику приймати рішення в межах своєї сфери відповідальності.

Перш ніж створити діаграму, спочатку необхідно перевірити точність і точність даних. Наприклад, якщо у вас є коефіцієнт конверсії 300%, як правило, від 50 до 70%, перевірте, звідки походить цей номер. Можливо, це було загальне поле, і ви мали всі дані разом. Можливо, це якась емісія, яка повинна бути вилучена з візуалізації, інакше цей стрибок знищить ціле зображення - 300% буде дорівнювати різниці між 50% і 70%. Через такий звіт у звіті ви можете помилитися і прийняти неправильне рішення.

аудиторії.

Створена система має два типи візуального відображення даних:

— 2D таблиця;
— 3D об’єкт.

Всі дані отримані с JSON-файлу система обробляє і надає точну схему поля. Спочатку опишемо 2D представлення: таблиця, яка має розмір поля. Максимальні координати беруться з файлу. Користувач має список глибин. Натиснувши на певну глибину – відкривається зафарбована таблиця. Кожний квадрат відповідає за певну координату. Кожний колір відповідає за щільність в даній точці. Якщо щільність 0 – квадрат буде білий. Якщо 100 – квадрат буде чорний.

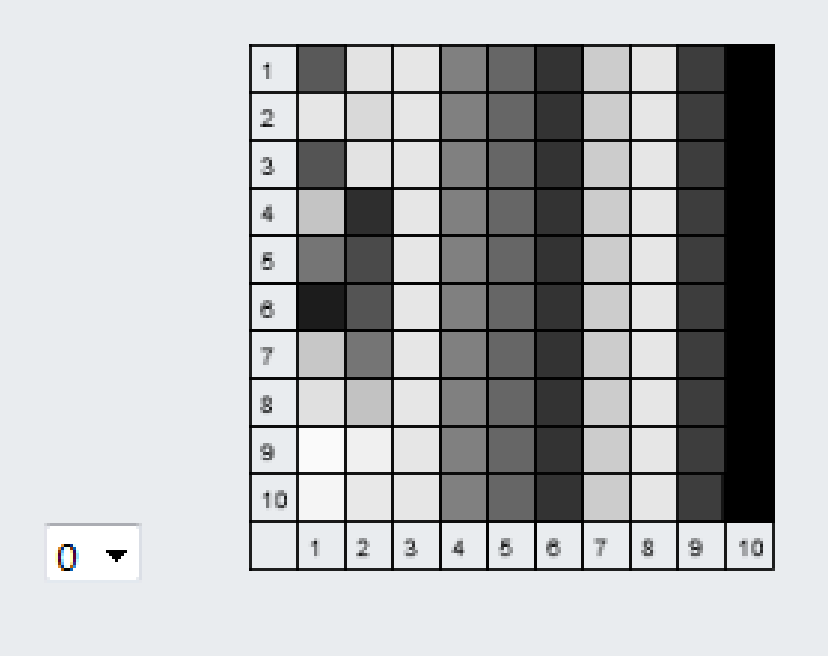


Рисунок 4.7 – 2D відображення даних.

Зрозуміло, що якщо ми маємо багато глибин, то користуватися 2D відображенням буде складно. Оскільки це відображення нам може показати стан щільності грунту в різних точках тільки на певній глибині.

Перевага повороту об'єкта. Просто переміщення об'єктів у кімнаті не дає додаткової переваги, якщо ви не маєте додаткової інформації про 2D-зображення. Проте потрібно обертати об'єкт (наприклад, аркуш паперу), тому що ви отримаєте зовсім інше зображення в 3D, ніж у 2D. У 2D це лист неправильної форми, і в 3D він залишається правильної форми, але розширюється в просторі.

У 2D важко зрозуміти їхнє положення, тоді як в 3D відносне розташування всіх об'єктів і їх кути обертання відразу зрозумілі.

Переваги співвідношення розмірів об'єктів (перспективи). У 2D-режимі принципи перспективи використовуються для створення ілюзії простору і співвідношення розміщення об'єктів (віддалені об'єкти є менш близькими

об'єктами, тінями, лініями, що сходяться до горизонту і т.д.), які не завжди точні дані можуть доставляти через об'єкти. Коли ви побачите об'єкт вперше, ви ніколи не визначите його розмір і позицію. У форматі 3D потрібно менше даних для співвідношення сторін об'єктів, і людина миттєво фіксує їх фактичний розмір і місце в просторі.

Навіть якщо об'єкти розташовуються в хаотичному порядку на різних відстанях. Спостерігач негайно визначає відстань до них і їх відносні розміри. На зображенні обидва об'єкти використовуються в просторі і видаляються з переглядача.

Більш інформативні окремі ділянки екрану (для складних об'єктів). У 3D-режимі складний об'єкт виглядає зрозумілим, тобто стопка графіки виглядає як чітка геометрична фігура. У режимі 2D складні геометричні конструкції не можуть використовуватися, оскільки вони не читаються. На малюнку показано лише два символи. Їх важко зрозуміти в 2D. Якщо ви подивитеся на них з 3D-анагліфними окулярами (синьо-червоним) (див. Малюнок), розберуть напис 3D. Це особливо очевидно в наукових проблемах (наприклад, складних хімічних сполуках), які не можна зрозуміти без 3D.

Таким чином, можна розмістити більш складну графіку в 3D режимі.

Використання нових форм діаграм. Стандартні 2D-схеми та графіка не доповнюють інформаційний вміст у 3D. У режимі 3D ви можете вставити додаткові змінні (не тільки одну змінну, але багато), не втрачаючи діаграм читабельності (див. Ілюстрацію в анагліфних окулярах (синьо-червоні)).

Це особливо корисно для тривимірної графіки, геоінформації та великих діаграм.

Те, що ви завжди вважали неглибоким, може стати об'ємним. Це дозволяє створювати нові візуальні фігури в 3D. Це дає вам можливість переглянути свій

погляд на життя і створити нові відчуття. Наприклад, ми завжди думали, що тінь рівна ...

Через це, система має змогу робити 3D відображення поля з усіма глибинами на одному зображенні.

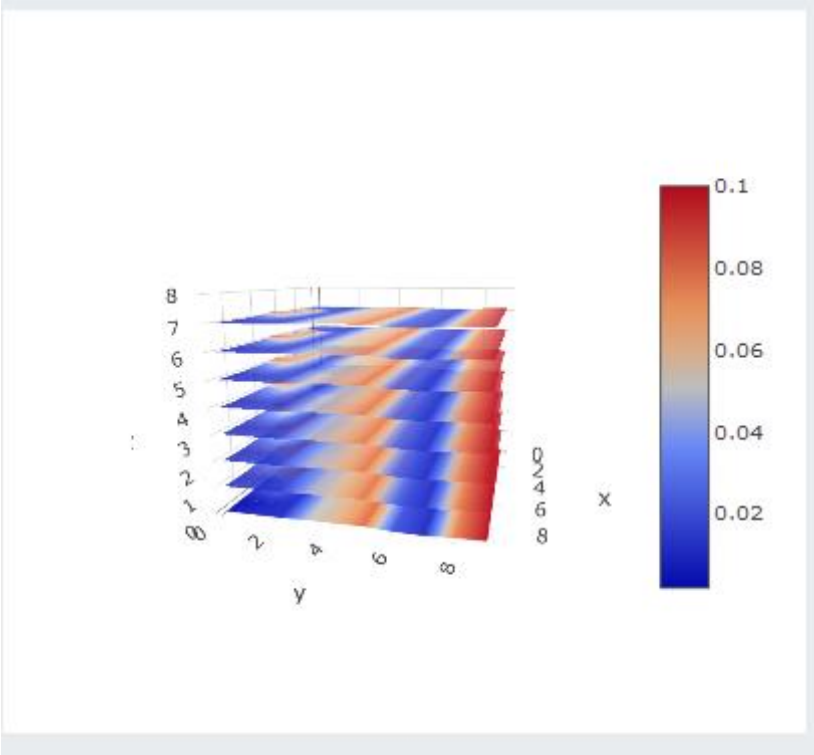


Рисунок 4.8 – 3D відображення даних

Аналізуючи 3D відображення, можна відмітити, що це дуже зручно для швидкого аналізу даних. Оскільки всі шари повністю видно на одному малюнку і не потрібно обирати різні глибини, як в 2D відображенні.

4.4 Звіт

Звіт - документ, який містить відомості про виконання завдань, доручень, планів і т.п. і представляється у вищестоящу організацію (посадовій особі).

Будь-який вид людської діяльності вимагає звітності.. Тому ми можемо підсумувати момент, коли нам потрібні звіти. Кожна людина несе відповідальність за справу, яку він робить, тому що він може незабаром вимагати звіту про прогрес, і що він може сказати, якщо він не має підзвітних документів ніколи в моєму, хоча і невеликий, але що хтось потребує життя..

Дана система може створювати звіти двох типів:

— Html сторінка;

— PDF файл.

Звіт у вигляді html сторінки є не дуже часто використовувані, але вони є зручними у використанні, якщо звіт не потрібно друкувати або зберігати. Тобто якщо звіт потрібно просто одноразово згенерувати і подивитися.

Користувач може створити звіт за двома напрямами:

— Звіт по аномальним точкам (де щільність від о до 10 і від 90 до 100);

— Звіт по точкам, які лежать в діапазоні, вказаному користувачем.

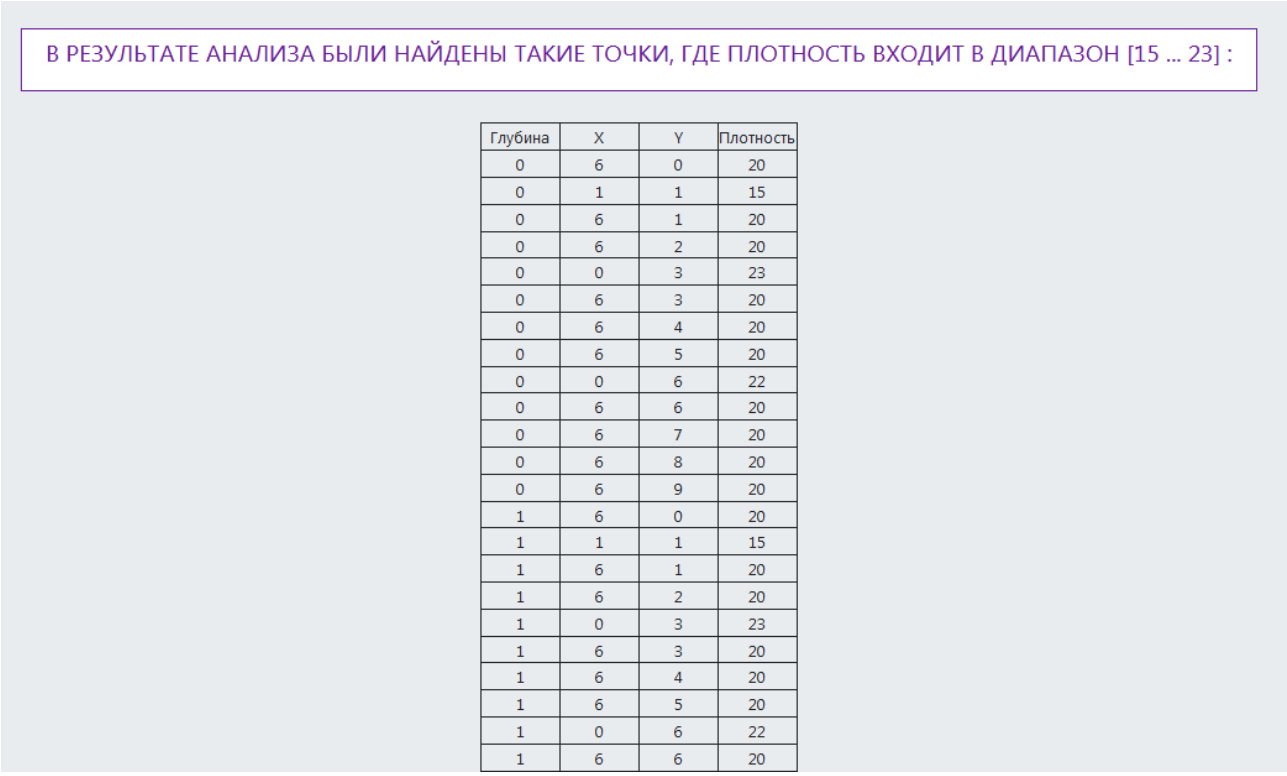
Такі звіти є дуже потрібно для якісного аналізу даних. Оскільки дуже часто потрібно швидко знайти ділянки поля, де щільність дуже низька або дуже висока. Натиснувши лише одну кнопку, ви зможете отримати список всіх точок на полі з аномальною щільністю.

В РЕЗУЛЬТАТЕ АНАЛИЗА БЫЛИ НАЙДЕНЫ АНОМАЛЬНЫЕ ТОЧКИ:

Глубина	X	Y	Плотность
0	2	0	10
0	7	0	10
0	9	0	100
0	0	1	10
0	2	1	10
0	7	1	10
0	9	1	100
0	2	2	10
0	7	2	10
0	9	2	100
0	2	3	10
0	7	3	10
0	9	3	100
0	2	4	10
0	7	4	10
0	9	4	100
0	2	5	10
0	7	5	10
0	9	5	100
0	2	6	10
0	7	6	10
0	9	6	100
0	2	7	10
0	7	7	10
0	9	7	100

Рисунок 4.9 – Звіт по аномальних точках

Також можливо таке, що користувачу потрібно знайти ділянки поля з особливою щільністю. Наприклад, від 30 до 35. Дана система має змогу робити звіт по точкам з діапазону, введеного користувачем, що значно полегшує процес проведення аналізу щільності грунту.



В РЕЗУЛЬТАТЕ АНАЛИЗА БЫЛИ НАЙДЕНЫ ТАКИЕ ТОЧКИ, ГДЕ ПЛОТНОСТЬ ВХОДИТ В ДИАПАЗОН [15 ... 23] :

Глубина	X	Y	Плотность
0	6	0	20
0	1	1	15
0	6	1	20
0	6	2	20
0	0	3	23
0	6	3	20
0	6	4	20
0	6	5	20
0	0	6	22
0	6	6	20
0	6	7	20
0	6	8	20
0	6	9	20
1	6	0	20
1	1	1	15
1	6	1	20
1	6	2	20
1	0	3	23
1	6	3	20
1	6	4	20
1	6	5	20
1	0	6	22
1	6	6	20

Рисунок 4.10 – Звіт по точкам з вказаного діапазону

Зрозуміло, що звіт у вигляді html файлу не можна роздрукувати або зберегти. Для цього, система має змогу відкривати звіт у вигляді pdf файлу.

Портативний формат документів (PDF) - це формат електронних документів з відкритим вихідним кодом, розроблений Adobe Systems з використанням різних функцій PostScript. По-перше, він призначений для презентації друкованої продукції в електронному вигляді. Є багато програм для перегляду, а також офіційна безкоштовна програма Adobe Reader. Значна кількість сучасного професійного друкарського обладнання підтримує формат PDF з апаратним забезпеченням, тому документи в цьому форматі можна друкувати без використання програмного забезпечення. Традиційним способом створення PDF-документів є наявність віртуального принтера. Сам документ створюється в спеціальній програмі - графічній програмі або текстовому редакторі, САПР тощо - і потім експортується в формат PDF для розповсюдження в електронному вигляді і на принтері.

У форматі PDF можна вставити необхідні шрифти (рядок за рядком), векторні та растрові зображення, форми та мультимедійні вставки. Підтримує RGB, CMYK, Grayscale, Lab, Duotone, Bitmap і різні типи стиснення растрових даних. Він має власні формати технічного друку: PDF / X-1a, PDF / X-3. Містить можливості електронного підпису для захисту та аутентифікації документів. Цей формат поширює велику кількість відповідних документів.

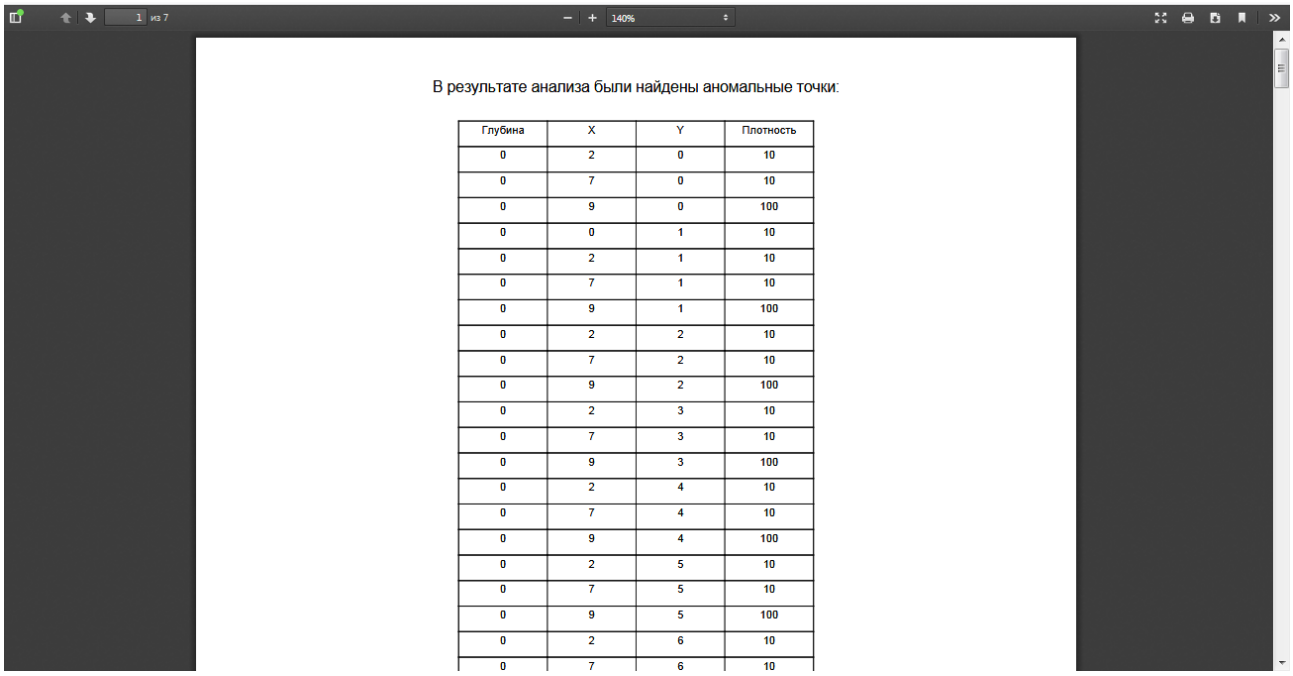


Рисунок 4.11 – Звіт у вигляді PDF.

Завдяки PDF формату, звіт можна зберегти або роздрукувати. Саме через це дуже багато звітів роблять у форматі pdf, тому що він зручний у використанні. Але для того, щоб відкрити файл такого формату – потрібно попередньо встановити редактор, який має змогу відкрити файл у форматі PDF.

Таку програму можна скачати в інтернеті, де їх велика кількість. І вже після того, як ви скачаєте цю програму, ви маєте змогу відкрити файл і робити будь-які дії, які можливі з файлом формату PDF.

4.5 Особистий кабінет

Особистий кабінет на сайті - це розділ, доступний тільки авторизованому користувачеві.

Багато послуг надають певні види послуг лише користувачам, які зареєструвалися з ними.

Для цього може бути декілька причин:

- Реєстраційні дані можна використовувати як готову базу для потенційних клієнтів для просування вашого продукту. Наприклад, багато постачальників програмного забезпечення пропонують безкоштовні пробні версії програмного забезпечення на певний період часу для отримання адреси електронної пошти. У майбутньому маркетологи будуть тісно співпрацювати з такою аудиторією.
- Обставини надання послуг, які вимагають дотримання певних законів і дій в рамках правової бази. Необхідною реєстрацією в даному випадку є підписання державних контрактів на пропозицію та інші документи.
- Конфіденційність особистої та фінансової інформації. Для операцій, що включають платіжні системи, системи реєстрації та бронювання, які вимагають надання персональних даних, послуга повинна обов'язково містити підтвердження прав.

Дана система має дуже простий у використанні особистий кабінет, який поділений на 4 частини. Завдяки цьому користувачу дуже легко орієнтуватися у власному особистому кабінеті, оскільки немає нічого зайвого.

Кабінет користувача розділений на 4 зони:

1) Ім’я користувача.
2) Зона, в якій знаходяться всі поля користувача.

3) Зона, де користувач створює власне поле.
4) Зона, де зберігаються звіти користувача.

Особистий кабінет відіграє дуже високу роль у забезпеченні цілісності даних і також забезпечує легку взаємодію користувача з системою.

Наприклад, при створенні звіту – звіт автоматично додається до особистого кабінету користувача, що значно полегшує пошук звітів.

При створенні нового поля користувач має заповнити два поля:

1) Назва поля;
2) JSON-файл, який містить дані про поле.

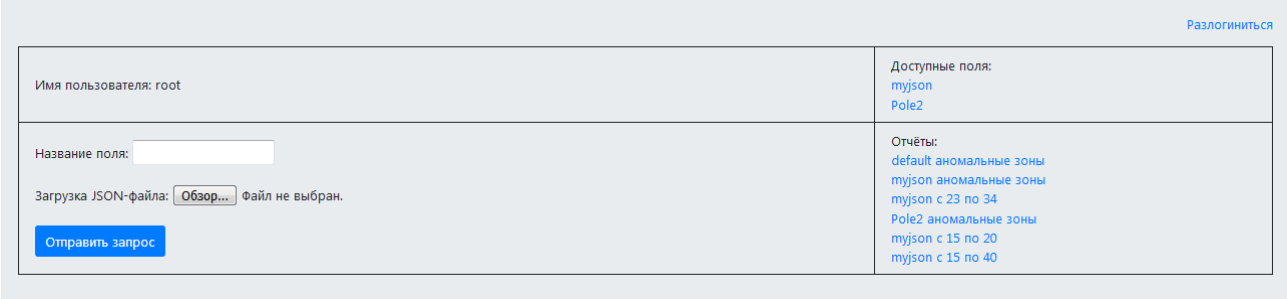


Рисунок 4.12 – Особистий кабінет

Також в особистому кабінеті користувача є кнопка Разлогиниться, яка допомагає користувачу вийти з системи, як авторизований користувач.

Висновки до розділу

В даному розділі було описано весь функціонал системи, проаналізовано архітектуру системи.

Основними функціями системи є:

— Авторизація;
— Візуальне відображення даних;
— Створення звітів.

5 ВЗАЄМОДІЯ КОРИСТУВАЧА З СИСТЕМОЮ

Інтерфейс користувача - елементи та компоненти програми, які можуть впливати на взаємодію користувача з програмним забезпеченням. в тому числі:

- засоби відображення інформації, відображуваної інформації, форматів і кодів;

- командні режими, мова інтерфейсу;

- пристрої та технології введення даних;

- діалоги, взаємодії та операції між користувачем і комп'ютером;

- Відгуки користувачів;

- Підтримка прийняття рішень в конкретній тематичній області;

- порядок використання програми та її документації.

Завдання користувача комп'ютерної програми - маніпулювати об'єктом та його властивостями - даними. На відміну від операторів, користувачі виконують професійне завдання з різною психологічною структурою дій, інших цілей, об'єктом праці і операцій, ресурсами, різним соціальним середовищем взаємодії. Різноманітність ситуацій, в яких можуть працювати інтерактивні програмні системи, ускладнює для розробника вибір цілей, які необхідно дотримуватися для створення успішного інтерфейсу. Різні дослідники та організації з розробки програмного забезпечення надають різні рекомендації, але головні з них такі:

Простота Ця рекомендація повертається до правила Оккама: найкраще пояснення є найпростішим. Дійсно, простий інтерфейс дозволяє користувачеві швидше адаптуватися, знижує ймовірність його помилок, а розробнику легше налагоджувати такий інтерфейс. Інтерактивна система хороша, якщо інтерфейс інтуїтивно зрозумілий, тобто він відповідає предметній області і стилю мислення користувача. Інтерфейс повинен бути легким в освоєнні, а не створювати бар'єр перед користувачем, який йому доведеться подолати, щоб потрапити на роботу.

Дружність (зручність використання) Інтерфейс зручний, якщо користувач, працюючи з ним, не відчуває дискомфорту. У користувача має скластися враження, що він керує процесом. Крім того, графічний інтерфейс

повинен бути побудований у відповідності з ергономічними вимогами: кольорами екрану і елементів, їх розміром, складом. Важливим є темп операцій, який повинен відповідати природному темпу людини, середньому часу відгуку та його дисперсії. Повідомлення повинні бути правильними за формою, точні та інформативні, неписьменні тексти абсолютно неприйнятні. Користувач повинен завжди знати, на якому етапі він знаходиться.

Природність інтерфейсу Природний інтерфейс - це той, що не змушує користувача істотно змінювати свої звичайні способи вирішення проблеми. Це, зокрема, означає, що повідомлення та результати, отримані заявкою, не повинні вимагати додаткових пояснень.

Функціональність Хоча обчислювальна система в деяких організаціях виконує роль великої іграшки, але найчастіше її намагаються використовувати для ведення бізнесу, особливо в тому випадку, коли виконання робіт іншими засобами менш ефективні. Функціональність системи свідчить про наявність значної ефективності при виконанні операцій, що робить її використання рентабельним. Інтерфейс повинен відображати його функціональність і дозволяти успішним користувачам різної кваліфікації працювати.

Розумна ціна Мова йде про виробничі системи. Зрозуміло, що система, яка має занадто дорогий інтерфейс, але недостатню функціональність, ймовірно, буде придбана, але користувач залишиться незадоволеним: термін окупності системи багато в чому залежить від функціональності. З іншого боку, економія на інтерфейсі є дуже недалекоглядною політикою. Погана якість інтерфейсу створює погану думку про систему про користувачів і може навіть призвести до відмови від її використання.

Людино-машинний інтерфейс - це термін, що включає технічні рішення, які дозволяють оператору взаємодіяти з машинами, якими він керує. Створення систем інтерфейсу людина-машина тісно пов'язане з ергономікою, але не ідентичною. Створити інтерфейс людина-машина, що охоплює: створення

робочого простору (крісла, столу або панелі керування), розміщення обладнання та засобів управління, освітлення робочого місця і, можливо, мікроклімату. Наступні дії оператора з органами контролю: їх наявність і необхідні зусилля, послідовність і "захист від дурнів", положення оголошень і розмір ярликів на них.

Взаємодія з комп'ютером людини (HCI) стосується проектування та використання комп'ютерних технологій на межі поділу між людьми (споживачами) та комп'ютерами. Дослідники в області HCI досліджують способи взаємодії людей з комп'ютерами і розробляють технології, які дозволяють людям взаємодіяти з комп'ютерами по-новому.

Інтерфейси є основою взаємодії в сучасних інформаційних системах. Якщо інтерфейс об'єкта (комп'ютера, програми, функції) не змінюється (стабільний, стандартизований), то можна змінювати сам об'єкт без зміни принципів взаємодії з іншими об'єктами. Це дозволяє масштабувати інтерфейс користувача та створювати інші пристрої або функції, які використовують цей тип взаємодії.

На ранніх етапах розвитку комп'ютерних технологій користувальницький інтерфейс розглядався як засіб спілкування з операційною системою і був досить примітивним. По суті, це дозволило виконувати певні дані, пов'язувати їх і виконувати деякі процедури обслуговування комп'ютера.

З часом з появою апаратних засобів з'явилася можливість створювати інтерактивне програмне забезпечення, що використовує користувальницькі інтерфейси користувача. В даний час основною проблемою є розробка інтерактивних інтерфейсів до складних програмних продуктів, розроблених для непрофесійних користувачів. В останні роки сформульовані основні концепції побудови таких інтерфейсів і запропоновано різні методи їх генерації.

Примітивний інтерфейс - організовує взаємодію з користувачем у консольному режимі. Як правило, такий інтерфейс реалізує певний скрипт програмного забезпечення.

На відміну від примітивного інтерфейсу користувача, користувач може вибрати необхідні операції зі спеціального списку, який відображається програмою в меню інтерфейсу користувача. Ці інтерфейси дозволяють реалізувати безліч робочих сценаріїв, порядок яких визначається користувачем.

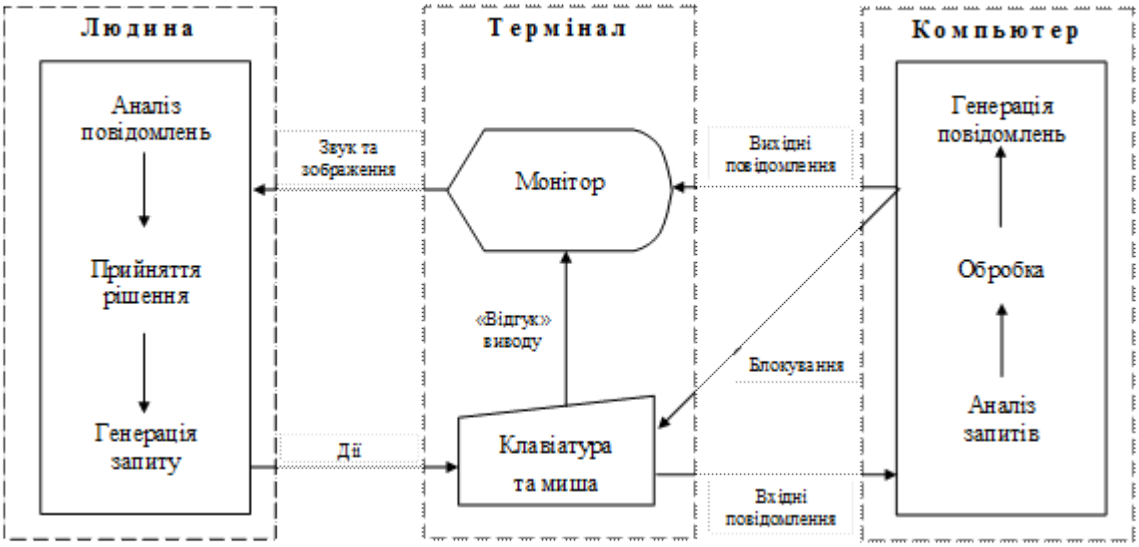


Рисунок 5.1 – Організація взаємодії користувача і комп’ютера

Розробка інтерфейсу включає ті самі основні кроки, що й розробка програмного забезпечення:

- завдання - визначення завдань інтерфейсу та загальних вимог до них;

- аналіз вимог та визначення специфікації - визначення сценаріїв використання та моделі інтерфейсу користувача;

- проектування - оформлення діалогів та їх реалізація у вигляді вхідно-вихідних процесів;

- впровадження - Програмування та тестування інтерфейсних процесів.

При проектуванні користувальницьких інтерфейсів необхідно враховувати психофізичні характеристики людини, пов'язані зі сприйняттям, пам'яттю і обробкою інформації, орієнтованої на мозок людини. Вивчення принципів людського мозку пов'язане з когнітивною психологією.

Інформація про зовнішній світ надходить у мозок у великих кількостях. Область мозку, умовно згадувана як «процесором сприйняття», постійно переставляє її без залучення свідомості, порівнюючи її з минулим досвідом, і вже зберігаючи її в пам'яті у вигляді візуальних, слухових і інших зображень. Будь-які раптові або значні зміни в людині в цьому районі привертають увагу. Інформація надходить у короткочасну пам'ять. Якщо не було звернуто уваги, інформація в сховищі прихована, а наступні частини замінені.

5.1 Системні вимоги

Системні вимоги - це опис наближених властивостей, яких повинен відповідати комп'ютер, щоб використовувати певне програмне забезпечення. Ці властивості можуть описувати апаратні вимоги (тип і частоту процесора, оперативну пам'ять, жорсткий диск) і програмне середовище (операційна система, наявність встановлених системних компонентів і послуг тощо). Зазвичай такі вимоги пред'являються виробником або автором програмного забезпечення.

Для деяких програм мінімальні та рекомендовані системні вимоги поділяються на:

1) Мінімальні системні вимоги - це набір умов, необхідних для запуску та експлуатації програмного продукту. Проте існування мінімальних системних вимог не впливає на здатність запускати програмне

забезпечення на комп'ютерах, які є більш слабкими, ніж мінімальні з точки зору продуктивності.

2) Рекомендовані системні вимоги - набір функцій, які передбачають оптимальне функціонування більшості функцій продукту. Навіть якщо комп'ютер відповідає рекомендованим системним вимогам, це не означає високу продуктивність програмного забезпечення. Наприклад, в деяких іграх неможливо грати з максимальними налаштуваннями графіки.

Для встановлення розробленої програмної системи персональний комп’ютер повинен мати процесор з тактовою частотою не нижче 2 GHz, на комп’ютері повинна бути встановлена операційна система Windows, MacOS або Linux. Для роботи з системою необхідний швидкий доступ до інтернету.

5.2 Інструкція користувача

Для початку роботи з системою, користувач повинен мати підключення до інтернету. Після цього, необхідно відкрити браузер через який і відбудеться вся взаємодія між системою і користувачем. Для переходу на дану систему, користувач має ввести певний url-адрес. На даний момент, він локальний і має вигляд: http://127.0.0.1:8000/ Після переходу по даному посиланню, відкривається головне меню програми.

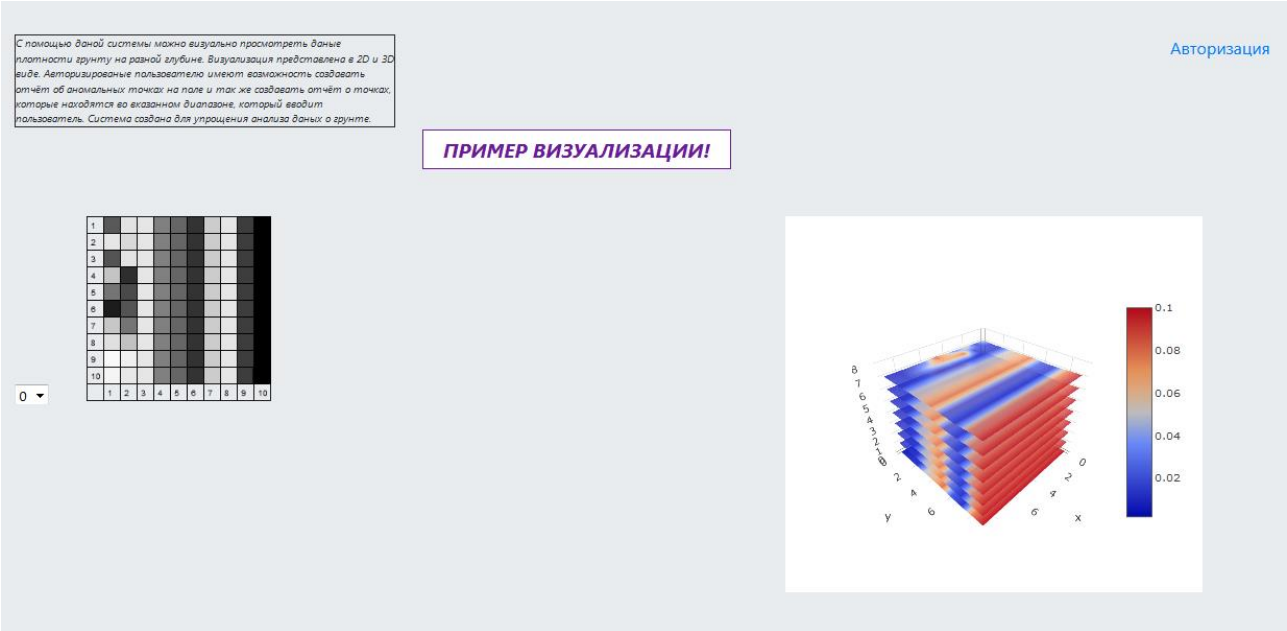


Рисунок 5.1 – Початкова сторінка системи

На початковій сторінці користувач бачить перед собою такі елементи інтерфейсу:

- Опис системи;
- Кнопка авторизації;
- Приклад візуалізації.

Хочу відмітити, що не авторизований користувач має право лише переглянути тестову версію системи. Тестова версія полягає в тому, що приклад візуалізації відбувається завжди по одному тестовому наборі даних. Це необхідно для того, щоб користувач подивився можливий функціонал і визначився, хоче він продовжувати подальшу взаємодію з системою чи ні.

Після того, як користувач оглянув початкову сторінку системи, він може або закрити систему, або нажати кнопку Авторизація. Після натиснення кнопки авторизація, користувачу відкривається нове вікно.

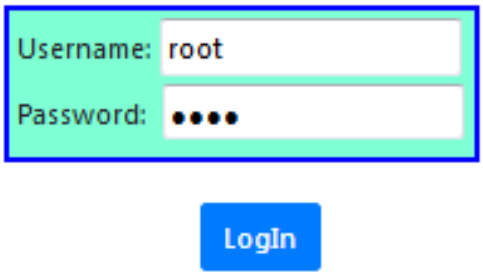


Рисунок 5.2 – Авторизація користувача

Після натиснення кнопки LogIn, користувач автоматично перенаправляється на головну сторінку, але вже в якості авторизованого користувача. Авторизація необхідна для забезпечення користувача додатковим розширеним функціоналом.Хочеться відмітити, що в системі відсутня самостійна реєстрація користувача. Щоб забезпечити власну безпеку підприємству, зареєструвати користувача в системі може лише довірена людина – адміністратор сайту. Завдяки використанню фреймворку Django, реєстрація відбувається в кабінеті адміністратора і має дуже зручний вигляд. Щоб увійти в кабінет користувача, потрібно створити superuser.

При реєстрації superuser потрібно заповнити три поля:

- Логін;
- Пароль;
- Email.

Після створення superuser, адміністратор може зайти у власний кабінет.

Маю зауважити, що кількість адміністраторів необмежена, але задля безпеки системи раціонально буде мати не більше 2х адміністраторів.

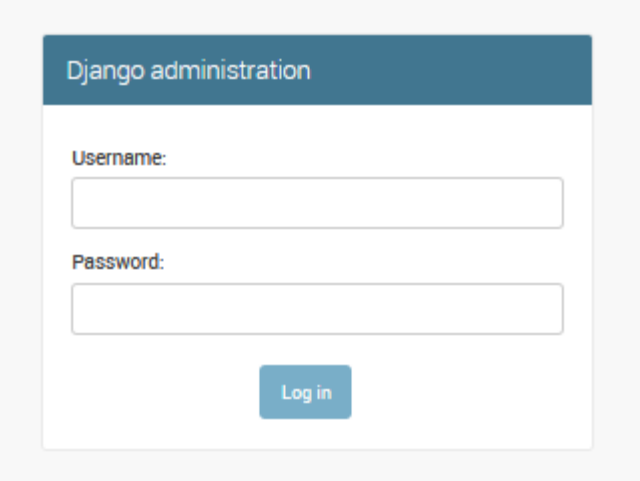


Рисунок 5.3 – Авторизація адміністратора

Після проведення авторизації, адміністратору відкривається головне вікно, де він може робити всі CRUD функції, такі як:

- Додавання користувача;
- Редагування даних про користувача;
- Видалення користувача.

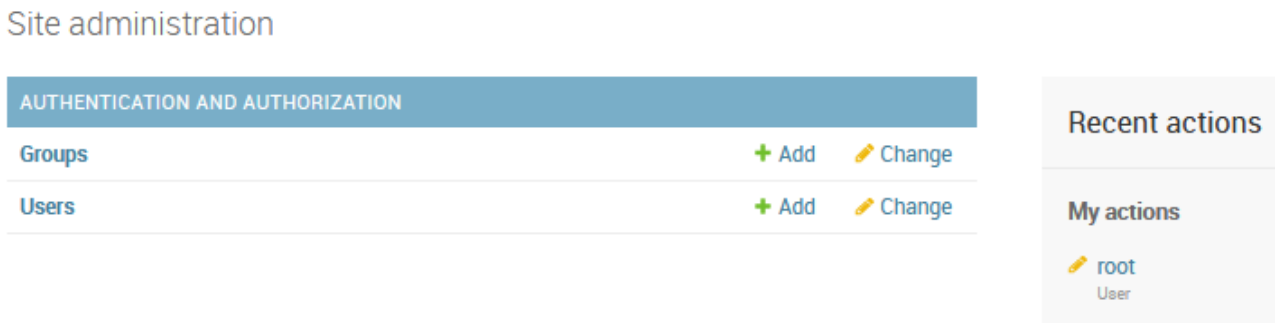


Рисунок 5.4 – Головна сторінка адміністратора

Щоб додати користувача, потрібно всього лиш натиснути кнопку Add і заповнити певні дані. Додавання користувача завдяки фреймворку Django відбувається дуже зручно і швидко завдяки вбудованій системі адміністрації. Адміністратор може зареєструвати необмежену кількість користувачів, тобто ліміту на кількість користувачів система не встановлений. З однієї сторони це

добре, оскільки не може такого бути, що користувач не зможе зареєструватися через перенавантаження системи. А з іншої сторони це погано, оскільки може бути багато користувачів, які лише одноразово зайшли в систему і тому дуже багато зайвих даних буде міститись в базі даних, що може значно пере загружати систему і уповільнювати її роботу.

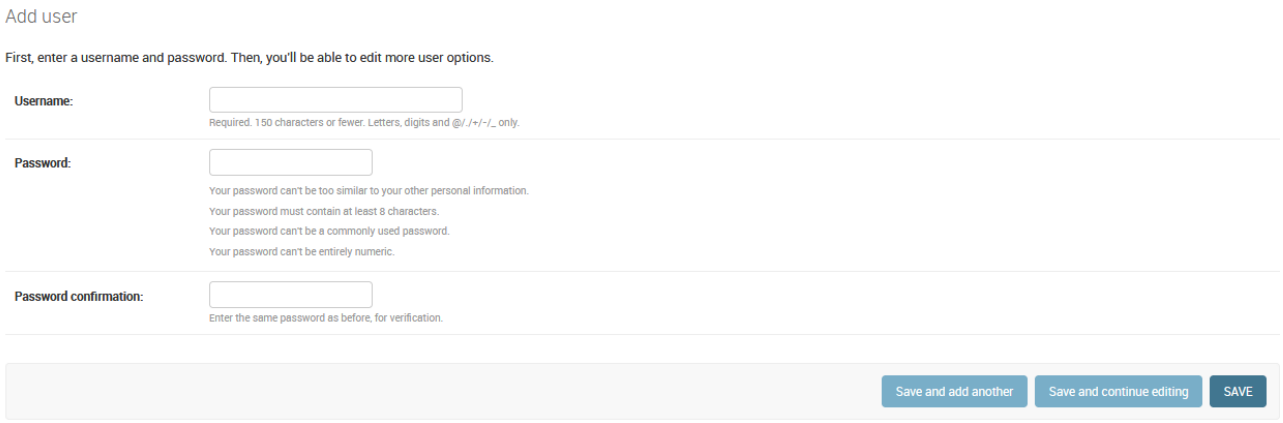


Рисунок 5.5 – Реєстрація користувача

Після ознайомлення з виглядом панелі адміністратора, повернемося до головного меню авторизованого користувача. Чим відрізняється головна сторінка не авторизованого користувача і авторизованого? Авторизований користувач має дещо розширений функціонал:

- Особистий кабінет;
- Створення звіту по діапазону;
- Створення звіту аномальних ділянок.

Авторизація дозволяє системі запам’ятати користувача. Оскільки, якщо користувач захоче візуалізувати дані по полю, які він робив раніше, йому вже не потрібно буде заново завантажувати JSON-файл. Тому що дане поле вже буде присутнє в його особистому кабінеті. Так само можна і сказати на рахунок звітів.

Як можна помітити з рисунка 5.6, головна сторінка авторизованого користувача дещо відрізняється від головної сторінки не авторизованого користувача.

По-перше можна помітити кнопку в правому верхньому кутку з написом Личный кабинет. Також під прикладом візуалізації маємо одне додаткове поле і дві кнопки для створення звітів.

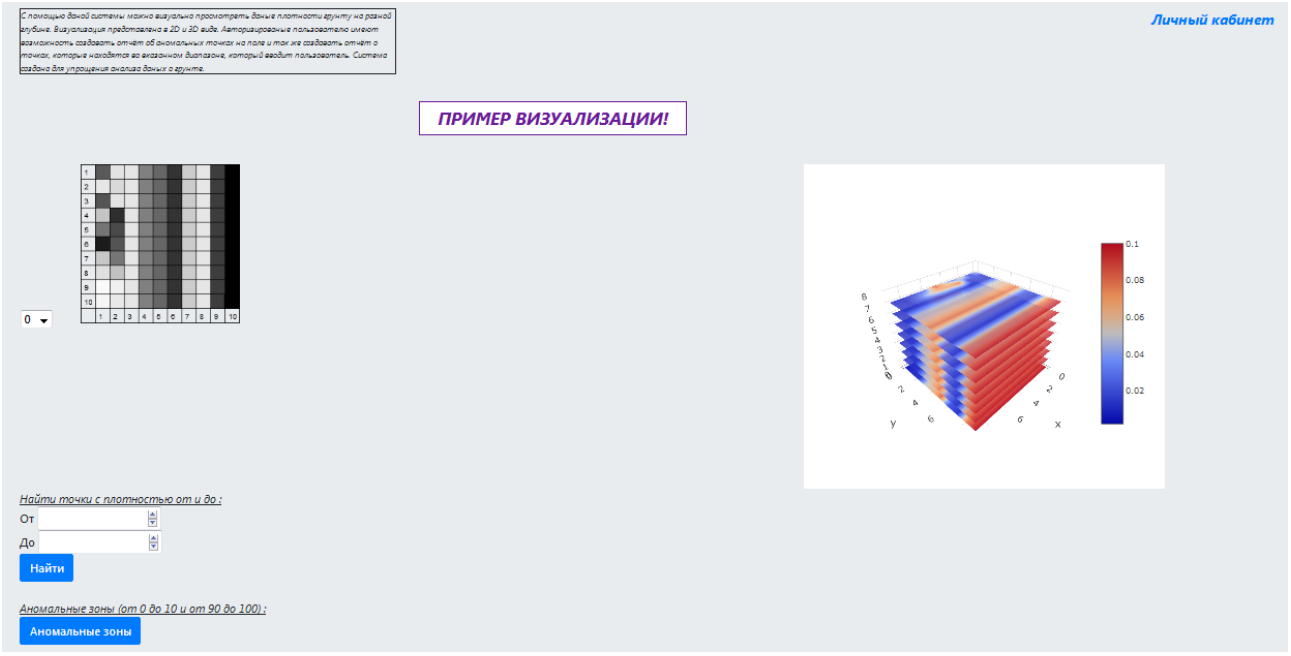


Рисунок 5.6 – Головна сторінка авторизованого користувача

В розділі 4 особистий кабінет уже був представлений у вигляді рисунка. Тому тут просто розберемо основний функціонал. Поділяється на 4 секції

В першій секції можна просто переглянути логін користувача, з якого ми перебуваємо в даній системі.

В другій секції ми можемо переглянути всі доступні нам поля. Оскільки користувач може робити візуалізацію поля не лише один раз, тому щоб значно скоротити витрати в часі, користувач може зберегти поле і в подальшому не треба буде знову створювати нове поле з таким же json- файлом.

В третій секції є можливість збереження даних. Для цього користувачу потрібно придумати назву для даного поля і загрузити, вибраний json-файл, який містить координати нашого поля. Відмітимо, що добавлення json-файлу можливе з будь-якого місця персонального комп'ютера. Тобто не важливо де зберігається ваш файл з даними. Він може навіть зберігатися або на флешці, або на диску.

Таким виглядом користувач може вибрати json-файл. За одну функцію створення користувач може строго створити лише одне поле, якому відповідає лише один унікальний json- файл.

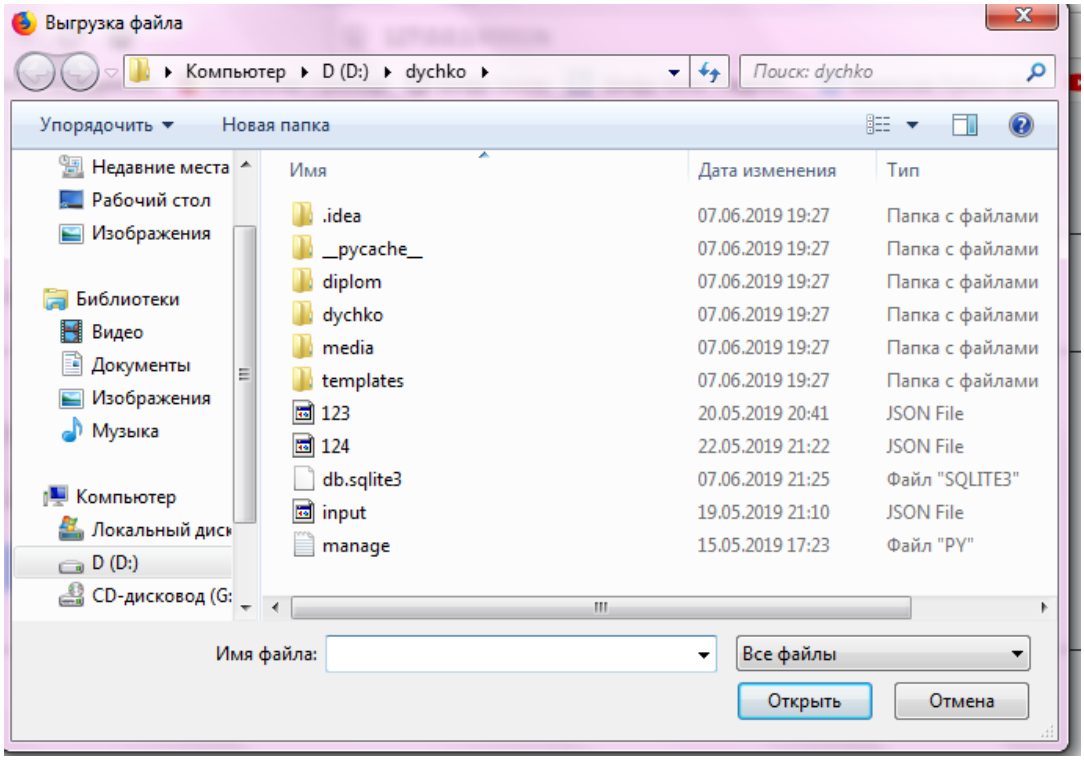


Рисунок 5.7 – Завантаження JSON –файлу.

В четвертій секції користувач має змогу переглянути всі збережені ним звіти. Це значно полегшує процес обробки і аналізу даних.

Також в особистому кабінеті присутня кнопка Разлогиниться, яка виходить с системи і направляє користувача до головної сторінки не авторизованого користувача.

Одним з основних моментів даної програми є створення звітів. Вони створюються в два типи:

— Html документ;

— PDF файл.

В першому варіанті система просто створює таблицю з усіма можливими розв’язками системи.

Але в другому варіанті користувач має шанс повністю створити справжній звіт з використання деяких додаткових даних.

До таких додаткових даних можна віднести:

— Час створення звіту;

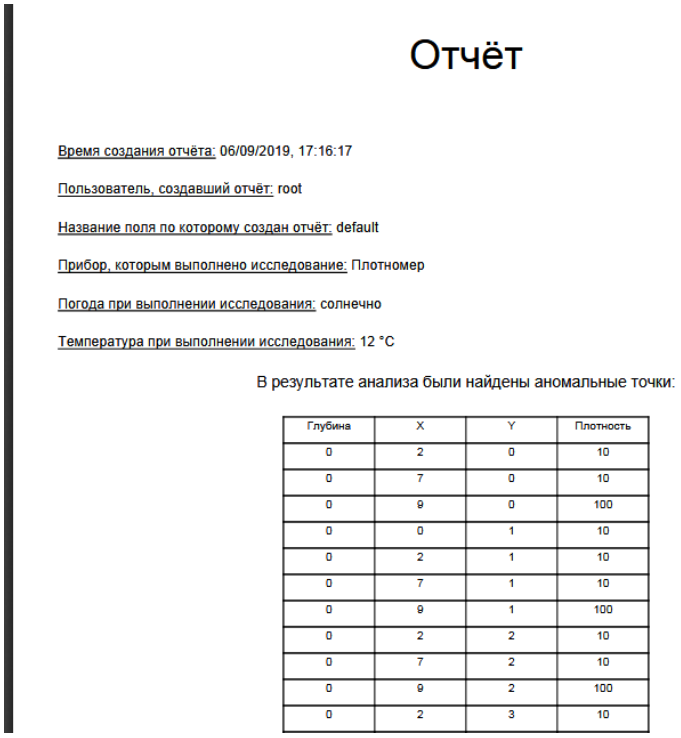
— Користувач, який створив звіт

— Назва поля по якому створено звіт;

— Прилад яким проведено аналіз;

— Погода при виконанні дослідження;

— Температура при виконанні дослідження.



Отчёт

Время создания отчёта: 06/09/2019, 17:16:17

Пользователь, создавший отчёт: root

Название поля по которому создан отчёт: default

Прибор, которым выполнено исследование: Плотномер

Погода при выполнении исследования: солнечно

Температура при выполнении исследования: 12 °C

В результате анализа были найдены аномальные точки:

Глубина	X	Y	Плотность
0	2	0	10
0	7	0	10
0	9	0	100
0	0	1	10
0	2	1	10
0	7	1	10
0	9	1	100
0	2	2	10
0	7	2	10
0	9	2	100
0	2	3	10

Рисунок 5.8 – Приклад звіту у вигляді PDF файлу

Всі функції перелічені вище є дуже важливими і необхідними для проведення швидко обробки даних і аналізу великих об’ємів інформації, і значно покращують якість проведення аналізу і зменшують час на проведення даного дослідження.

ВИСНОВКИ

Розроблена система відповідає висуненим вимогам. Система здатна працювати із значними обсягами даних з високою швидкістю. Також система містить зручний інтерфейс для користувача.

Було використано декілька технологій для написання програми, зокрема:

— Python;

— Django;

— HTML/CSS.

Систему було протестовано на декількох різних наборах вхідних даних.

Програмний продукт полегшив процес праці робітників, чия робота пов’язана з аналізом значних об’ємів даних. Систему можна використовувати не тільки в аграрній сфері, але й там де важлива така характеристика грунту, як щільність.

Система повністю готова до використання і впровадження на аграрному підприємстві.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Sweigart Invent Your Own Computer Games with Python. - 2008-2010. - 436 с.

2. Марк Лутц. Программирование на Python, 4-е видання, II том - Переклад з англійської. - СПб.: Символ-Плюс, 2011.

3. Марк Лутц. Программирование на Python, 4-е видання, I том - Переклад з англійської. - СПб.: Символ-Плюс, 2011. - 992 с

4. Марк Лутц. Изучаем Python, 4-е видання. - Переклад з англійської. - СПб.: Символ-Плюс, 2010. - 1280 с у

5. Дэвид М. Бизли. Python. Подробный справочник, 4-е видання. - Переклад з англійської. - СПб.: Символ-Плюс, 2010. - 864 с

6. Ноа Гифт, Джереми М. Джонс. Python в системном администрировании UNIX и Linux. - Переклад з англійської. - СПб.: Символ-Плюс, 2009. - 512 с

7. Бизли, Дэвид М. Язык программирования Python. Справочник. - К.: ДиаСофт, 2000. - 336 с.

8. Сузи, Р.А. Python. Наиболее полное руководство (+CD). - СПб.: БХВ-Петербург, 2002. - 768 с.

ДОДАТОК 1

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51147_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51147_19Б 81-1	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51147_19 Б 12-1	dychko	Папка з програмою
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТВ51147_19 Б 13-1	Довідка.doc	Довідка про програму

ДОДАТОК 2

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ АПЕПС_ТВ51147_19Б 12-1

Аркушів 9

Київ 2019

View.py

```
import datetime

import json

import os

from datetime import datetime
```

```
from io import BytesIO
from random import choice

import xhtml2pdf.pisa as pisa
from django.conf import settings
from django.http import HttpResponse
from django.shortcuts import redirect
from django.shortcuts import render
from django.template.loader import get_template

from .models import Data, Otchet



def create_arr(size, init_val):
    arr = [0] * size
    for i in range(size):
        arr[i] = ([0] * size)
        for j in range(size):
            arr[i][j] = init_val
    return arr



def index_view(req, pk=None):
    try:
        data = Data.objects.filter(name=pk).first()
        json_id = data.id
        data = data.data
        data = json.loads(data)
    except Exception as ex:
        file = open('input.json', 'r')
```

```
    data = json.load(file)['data']
    file.close()
    obj, created = Data.objects.get_or_create(
        data=data,
        defaults={'user': None, 'name': 'default'}
    )
    json_id = obj.id if obj else created.id

max_width = 0
for z in data:
    for x, y, d in z:
        max_width = max(x, y, max_width)
max_width += 1



data_3d = []
data_2d = []

for z_index, z in enumerate(data):
    output_3d = create_arr(max_width, z_index)
    for x, y, d in z:
        output_3d[x][y] += d / 1000
    data_3d.append(output_3d)
    data_2d.append(z)

context = {
    'data_3d': data_3d,
    'data_2d': data_2d,
    'z_count': range(len(data_2d)),
    'width': max_width,
```

```
        'json_id': json_id,


    }
    return render(req, 'index.html', context)



def login_view(req):
    return render(req, 'registration/login.html')



def lk_view(req):
    if not req.method == 'POST':
        datas = Data.objects.filter(user=req.user)
        otchets = Otchet.objects.filter(user=req.user)

        context = {
            'datas': datas,
            'otchets': otchets
        }
        return render(req, 'lk.html', context)

    name = req.POST['name']
    try:
        file = req.FILES['json'].read()
        data = json.loads(file)['data']
    except:
        return render(req, 'lk.html')

    obj, created = Data.objects.get_or_create(
        data=data,
```

```
        defaults={'user': req.user, 'name': name}
    )

    name = obj.name if obj else created.name
    return redirect('/' + name)




def otchet_view(req):
    f = t = 0
    as_file = req.GET.get('as_file', False)

    try:
        anomal = bool(req.GET.get('anomal', False))
        if anomal:
            func = lambda d: (0 <= d <= 10 or 90 <= d <= 100)
        else:
            f = int(req.GET['from'])
            t = int(req.GET['to'])
            func = lambda d: (f <= d <= t)

        json_id = int(req.GET['json_id'])
        data_obj = Data.objects.filter(id=json_id).first()
        data = json.loads(data_obj.data)

        obj, created = Otchet.objects.get_or_create(
            json_id=json_id,
```

```
        anomal=anomal,
        f=f,
        t=t,
        user=req.user,
    )

except Exception as ex:
    return redirect('/')

answ = []
for z, zd in enumerate(data):
    for x, y, d in zd:
        if func(d):
            answ.append(
                (x, y, z, d)
            )

context = {
    'dots': answ,
    'from': f,
    'to': t,
    'anomal': anomal,
    'time': datetime.now().strftime("%m/%d/%Y, %H:%M:%S"),
    'user': req.user.username,
    'name': data_obj.name,
    'weather': choice(['солнечно', 'облачно', 'дождливо']),
    'temp': choice([10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30])
}

print(context)
```

```
    if as_file:
        template = get_template('invoice.html')
        html = template.render(context)
        response = BytesIO()
        pdf = pisa.CreatePDF(BytesIO(html.encode("UTF-8")), response, encoding='UTF-8',
link_callback=fetch_pdf_resources)
        if not pdf.err:
            return HttpResponse(response.getvalue(), content_type='application/pdf')
        else:
            return HttpResponse("Error Rendering PDF", status=400)

    return render(req, 'otchet.html', context)



def fetch_pdf_resources(uri, rel):
    if uri.find(settings.MEDIA_URL) != -1:
        path = os.path.join(settings.MEDIA_ROOT, uri.replace(settings.MEDIA_URL, ''))
    elif uri.find(settings.STATIC_URL) != -1:
        path = os.path.join(settings.STATIC_ROOT, uri.replace(settings.STATIC_URL, ''))
    else:
        path = None
    return path
```

Models.py

```
from django.db import models
```

```
from django.conf import settings

# Create your models here.



class Data(models.Model):
    name = models.CharField(max_length=200, default='')
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, null=True)
    data = models.CharField(max_length=1200, unique=True)



class Otchet(models.Model):
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, null=True)
    json_id = models.IntegerField()
    anomal = models.BooleanField(default=False)
    f = models.IntegerField(null=True)
    t = models.IntegerField(null=True)

    def __str__(self):
        data_name = Data.objects.filter(id=self.json_id).first()
        name = data_name.name
        print(data_name.name)
        if self.anomal:
            name += " аномальные зоны"
        else:
            name += f" с {self.f} по {self.t}"

        return f"<a href='/otchet?anomal={self.anomal}&json_id={self.json_id}" \
            f"&f={self.f}&t={self.t}'>{name}</a>"
```

Urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf.urls import url
from diplom import views
from django.conf.urls.static import static
from django.conf import settings



urlpatterns = [
    path('admin/', admin.site.urls),
    path('lk', views.lk_view, name='lk'),
    path('otchet', views.otchet_view),
    url(r'^accounts/', include('django.contrib.auth.urls')),
    path('<str:pk>', views.index_view, name='base'),
    path('', views.index_view, name='base'),
]
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
```

```
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

</head>

<style>


 .header-h1 h1 {

  display: inline-block;

  position: absolute;

  margin-left: 30%;

  margin-top: 7%;

  background: #fff;

  margin-bottom: 0;

  font-size: 1.5rem;

  text-transform: uppercase;

  padding: .5rem 1.5rem;

  border: .125rem solid #6a1b9a;

  color: #6a1b9a;

 }

 .header-h1 h1::after {

  content: "";

  position: absolute;

  background: #6a1b9a;

  height: .125rem;

  left: 0;

  top: 50%;

  width: 100%;

  transform: translateY(-50%);

  z-index: -999;

 }

 .header-h1-dark h1 {
```

```
    background: #6a1b9a;
    color: #fff;
}
   .first {
    position: absolute;
     margin-left: 85%;
    font-size: 20px;



   }
   .second {
    font-size: 12px;
    font-style: italic;
    margin-right: 70% ;
    position: absolute;
    border: solid 2px;

   }

   .third {
    margin-top:40%;
    position: absolute;
    border: solid 2px;
    font-size: 20px;



   }
  }
</style>
<body>
```

```
<div class="jumbotron">
    <div class="header-h1">

    <div class="first">
        {% if user.is_authenticated %}
     <a href="/lk"><i><strong>Личный кабинет</strong></i></a>

        {% else %}
         <a href="/accounts/login/"><i><strong>Авторизация</strong></i></a>

        {% endif %}
    </div>
    <div class="second">
```

С помощью даной системы можно визуально просмотреть даные плотности грунту на разной глубине. Визуализация представлена в 2D и 3D виде.

Авторизированые пользователю имеют возможность создавать отчёт об аномальных точках на поле и так же создавать отчёт о точках, которые находятся во вказанном диапазоне, который вводит пользователь. Система создана для упрощения анализа даных о грунте.

```
    </div>
     <h1><i> <strong> Пример визуализации! </strong> </i></h1>

    <table width="100%">
    <tr>
        <td width="50%" margin-left=50px valign="top">
            <br> <br> <br> <br><br> <br> <br> <br><br>
            <select onchange="Drawing(this.value)">
                {% for i in z_count %}
                    <option>{{ i }}</option>
                {% endfor %}
            </select>
                   
```

```
            <canvas id="canvas"></canvas>
        </td>
        <td width="50%" align="center">
            <br><br><br><br><br><br> <br> <br> <br>
            <div id="myDiv" style="width:500px; height:60%;"></div>
        </td>
    </tr>
</table>
</div>



{% if user.is_authenticated %}



    <i><u>Найти точки с плотностью от и до : </u></i>
    <form action="otchet">
        {% csrf_token %}
        <input type="hidden" name="json_id" value="{{ json_id }}">
        От <input type="number" name="from"> <br>
        До <input type="number" name="to"> <br>
        <input type="submit" value="Найти" class="btn btn-primary">
    </form>
    <br>
   <i><u> Аномальные зоны (от 0 до 10 и от 90 до 100) :</u></i>
    <form action="otchet">
        {% csrf_token %}
        <input type="hidden" name="json_id" value="{{ json_id }} ">
        <input type="submit" name="anomal" value="Аномальные зоны " class="btn btn-primary">
    </form>
{% else %}
```

```
    <br><br><br><br><br><br><br><br>

    {% endif %}



<script>

    canvas = document.getElementById('canvas');
    ctx = canvas.getContext("2d");

    cell_size = 20;
    width = {{ width }}+1;

    canvas.width = cell_size * width + 1;
    canvas.height = cell_size * width + 1;

    data_2d = {{ data_2d|safe }};

    function Drawing(depth) {
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        for (i of data_2d[depth]) {
            rect(i[0], i[1], i[2]);
        }
        draw_captions();
    }

    function rect(x, y, color) {
        x += 1;
        color = 255 - (color / 100 * 255);
        ctx.beginPath();
```

```
    ctx.rect(x*cell_size, y*cell_size, cell_size, cell_size);
    ctx.closePath();
    ctx.fillStyle = 'rgb('+color+','+color+','+color+')';
    ctx.fill();
}


function draw_captions() {
    for (i=0; i<width+1; i++) {
        l = i * cell_size;
        w = width * cell_size;
        ctx.moveTo(0.5, l+0.5);
        ctx.lineTo(w+0.5, l+0.5);
        ctx.moveTo(l+0.5, 0.5);
        ctx.lineTo(l+0.5, w+0.5);
        ctx.stroke();

        if (i != width-1) {
            draw_text(i+1, width, i + 1);
            draw_text(0, i+1, i + 1);
        }
    }
}


function draw_text(x, y, text) {
    ctx.fillStyle = 'rgb(0,0,0)';
    ctx.fillText(text, (x+0.25)*cell_size, (y-0.25)*cell_size);
}
```

```
Drawing(Object.keys(data_2d)[0]);
```

```
data_3d = {{ data_3d|safe }};
plt = [];

plt.push({z: data_3d[0], type: 'surface'});

for (i=1; i<data_3d.length; i++)
    plt.push({z: data_3d[i], type: 'surface', opacity: 0.9, showscale: false});


var layout = {
  scene: {
      zaxis: {
          range: [0, data_3d.length],
      },
  }
};

Plotly.newPlot('myDiv', plt, layout);
```

```
</script>


</div>
</body>
</html>
```

Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys



def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'dychko.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)



if __name__ == '__main__':
    main()
```

ДОДАТОК 3

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ АПЕПС_ТВ51147_18Б 13-1

Аркушів 7

Київ 2019

АНОТАЦІЯ

В даному додатку можна знайти опис основного компоненту аграрної системи для аналізу щільності грунту, що виконує певні із завдань, поставлених в розділі 1, а саме:

- отримання даних в форматі JSON файлу;
- 2D і 3D візуалізація вхідних даних;
- формування звіту про аномальні точки на вибраному полі;
- формування звіту точок по вказаному діапазону щільності від користувача;

Дані система отримує у вигляді JSON файлу.

Компонент розроблений з використанням мови програмування Python у середовищі програмування PyCharm і також з використанням технології Canvas і HTML.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

В даному додатку можна знайти опис основного компоненту аграрної системи для аналізу щільності грунту, що виконує певні із завдань, поставлених в розділі 1. У додатку 2 міститься частина програмного коду компоненту.

Система працює в операційних системах Windows 7, Windows 8, Windows 10, Linux і MacOS і потребує встановлення браузеру та доступу до інтернету.

Компонент розроблений з використанням мови програмування Python у середовищі програмування PyCharm і також з використанням технології Canvas і HTML.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Компонент виконує певні завдання, пов’язані з аналізом щільності грунту:

- отримання даних в форматі JSON файлу;
- 2D і 3D візуалізація вхідних даних;
- формування звіту про аномальні точки на вибраному полі;
- формування звіту точок по вказаному діапазону щільності від користувача;
- збереження всіх доданих файлів JSON і сформованих звітів до кожного поля.

Точність аналізу і загальне відображення щільності грунту залежить від масштабів поля і від кількості вхідних точок.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для встановлення розробленої програмної системи персональний комп’ютер повинен мати процесор з тактовою частотою не нижче 2 GHz, на комп’ютері повинна бути встановлена операційна система Windows 7, Windows Vista SP2, Windows XP SP2, Windows Server 2008 SP2, Windows Server 2003 SP2. Також на жорсткому диску повинно бути не менше 10 Мб вільного місця.

Для роботи з системою необхідний швидкий доступ до інтернету.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Розроблена система не потребує додаткового завантаження. Головне – це доступ до інтернету. Після виконання входу в інтернет через браурез, потрібно перейти по локальному посиланню і зайти в систему.